PB85-173946

Optimization of Nitrogen Removal by Rapid Infiltration. Volume 2 Remote Computer Operating System

Colorado Univ. at Boulder

Prepared for

Robert S. Kerr Environmental Research Lab.
Ada, OK

Mar 85

U.S. Department of Commerce
National Technical Information Service
NTIS

PB85-173946

EPA/600/2-85/016b
March 1985

OPTIMIZATION OF NITROGEN REMOVAL BY RAPID INFILTRATION

VOLUME II: REMOTE COMPUTER OPERATING SYSTEM

by

David M. Walters
Carl G. Enfield
Robert S. Kerr Environmental Research Laboratory
U.S. Environmental Protection Agency
Ada, Oklahoma 74820

and

Edwin R. Bennett
Department of Civil, Environmental
and Architectural Engineering
University of Colorado
Boulder, Colorado 80309

Grant No. 807933

Project Officer

Lowell E. Leach
Robert S. Kerr Environmental Research Laboratory
U.S. Environmental Protection Agency
Ada, Oklahoma 74820

ROBERT S. KERR ENVIRONMENTAL RESEARCH LABORATORY
OFFICE OF RESEARCH AND DEVELOPMENT
U.S. ENVIRONMENTAL PROTECTION AGENCY
ADA, OK 74820

REPRODUCED BY
NATIONAL TECHNICAL
INFORMATION SERVICE
U.S. DEPARTMENT OF COMMERCE
SPRINGFIELD, VA. 22161

TECHNICAL REPORT DATA
(Please read Instructions on the reverse before completing)

1. REPORT NO.
EPA/600/2-85/016b

2.

3. RECIPIENT'S ACCESSION NO.
PB85 173946 /AS

4. TITLE AND SUBTITLE
OPTIMIZATION OF NITROGEN REMOVAL BY RAPID INFILTRATION
Volume II: Remote Computer Operating System

5. REPORT DATE
March 1985

6. PERFORMING ORGANIZATION CODE

7. AUTHOR(S)
D. M. Walters, C. G. Enfield, and E. R. Bennett

8. PERFORMING ORGANIZATION REPORT NO.

9. PERFORMING ORGANIZATION NAME AND ADDRESS
University of Colorado, Dept. of Civil, Environmental
and Architectural Engineering, Boulder, CO 80309
and
U.S. Environmental Protection Agency
P.O. Box 1198, Ada, OK 74820

10. PROGRAM ELEMENT NO.
CAZB1B

11. CONTRACT/GRANT NO.
CR-807933

12. SPONSORING AGENCY NAME AND ADDRESS
U.S. Environmental Protection Agency
Robert S. Kerr Environmental Research Laboratory
P. O. Box 1198
Ada, OK 74820

13. TYPE OF REPORT AND PERIOD COVERED
Final 9/22/80 - 12/21/84

14. SPONSORING AGENCY CODE
EPA-600/15

15. SUPPLEMENTARY NOTES

Project Officer: Lowell E. Leach

16. ABSTRACT

A rapid infiltration municipal wastewater land treatment system using two infiltration beds was remotely controlled in Boulder, Colorado, utilizing telecommunication from the U.S. Environmental Protection Agency, Robert S. Kerr Environmental Research Laboratory in Ada, Oklahoma, with a 16-bit computer system. Operation of the system required unique interfacing of a PDP1123 computer system with a data collection system and signal conditioning system. When this hardware was integrated together, it allowed data to be collected at Boulder, Colorado, and the results transmitted to the RSKERL in Ada, Oklahoma, where computer programs analyzed the data and transmitted codes back to Boulder to operate the system as desired.

17. KEY WORDS AND DOCUMENT ANALYSIS

a. DESCRIPTORS	b. IDENTIFIERS/OPEN ENDED TERMS	c. COSATI Field/Group
Land use Ground water Purification Quality control Sewage treatment Nutrient removal	Land application High-rate application Primary pretreatment Remote computer control Nitrogen control Sewage effluents	68D 48B, E, G

18. DISTRIBUTION STATEMENT RELEASE TO PUBLIC	**19. SECURITY CLASS** *(This Report)* UNCLASSIFIED	**21. NO. OF PAGES** 336
	20. SECURITY CLASS *(This page)* UNCLASSIFIED	**22. PRICE**

EPA Form 2220-1 (Rev. 4-77) PREVIOUS EDITION IS OBSOLETE

DISCLAIMER

The information in this document has been funded wholly or in part by the United States Environmental Protection Agency under assistance agreement CR807933 to the University of Colorado. It has been subjected to the Agency's peer and administrative review and has been approved for publication as an EPA document. Mention of trade names or commercial products does not constitute endorsement or recommendation for use.

FOREWORD

The U.S. Environmental Protection Agency was established to coordinate the administration of major Federal programs designed to protect the quality of our environment.

An important part of the Agency's effort involves the search for information about environmental problems, management techniques, and new technologies through which optimum use of the Nation's land and water resources can be assured and the threat pollution poses to the welfare of the American people can be minimized.

The U.S. Environmental Protection Agency's Office of Research and Development conducts this search through a Nationwide network of research facilities. As one of these facilities, the Robert S. Kerr Environmental Research Laboratory is responsible for the management of programs including the development and demonstration of soil and other natural systems for the treatment and management of municipal wastewaters.

The rapid infiltration treatment process for land treatment of municipal wastewaters uses the unsaturated soil profile beneath infiltration basins as the treatment media. This project was designed to evaluate the optimization of nitrogen removal in the rapid infiltration system using primary wastewater from the Seventy-Fifth Street Wastewater Treatment Plant in Boulder, Colorado. Two previous studies conducted at the same site evaluated the treatment efficiency for maximum hydraulic loading of applied secondary and primary pretreated municipal wastewater. Results demonstrate that high rate systems can be cost effective alternatives for municipal sewage treatment at sites where conditions are favorable for high hydraulic loading and an excess of seventy percent of total nitrogen removal can be achieved.

This report contributes to the knowledge which is essential for the U.S. Environmental Protection Agency to meet requirements of environmwental laws and enforce pollution control standards which are reasonable, cost effective, and provide adequate protection for the American public.

Clinton W. Hall

Clinton W. Hall
Director
Robert S. Kerr Environmental
Research Laboratory

ABSTRACT

A rapid infiltration wastewater treatment system was remotely controlled in Boulder, Colorado utilizing telecommunication lines from Ada, Oklahoma with a 16-bit computer system. Environmental and operating parameters were collected in Boulder and then transmitted to Ada where computer software determined the schedule and duration of wastewater applications. Temperature corrected soil conductivity was utilized in the control of soil moisture levels. Minimum air temperature and maximum wind speed were monitored to prevent pipe freezing and inadvertent off-site distribution. The control plots could be dynamically changed as the system operated to compensate for changing environmental conditions. Soil moisture levels to control the system could also be dynamically altered. Reference resistors were used to monitor the reliability of the system.

TABLE OF CONTENTS

Disclaimer ii
Foreword iii
Abstract iv
List of Figures vii
List of Tables viii

1. Introduction 1
2. Conclusions 1
3. Hardware Components 2
    Computer system 2
    Data collection system hardware 8
    Signal conditioning interfaces 14
    Temperature interface 14
    Conductivity interface 18
    Voltage-to-frequency interface 19
    Pump control interface 23
    Flow measurement 23
    Anemometer interface 23
4. Computer Software 29
    Introduction 29
    System software 29
    Nonsystem distributed software 29
    Configured or system generated system 30
    Software control of the computer calling unit 30
    Software control of the hp data collection system . . . 33
    Sampling data at remote site 38
    Foreground - background interaction 40
    Internal traceback feature 43
    Land Treatment Control and Monitoring Programs 44
        Introduction 44
        Control Program (BOULD.REL) 44
            Main driver 45
            Main subroutines 45
            Secondary and control subroutines 50
        Background Monitor Program (BOURST.SAV) 51
            Introduction 51
            MAIN DRIVER 51
            MAIN SUBROUTINES 52
        File Maintenance Program (BOUFIL.SAV) 52
            Main driver 52
            Main subroutines 52
        Timlib 53
        Video routines 54
        Command files to build and operate system 54

References 56

Appendix I. Control program source codes 58
Appendix II. Monitoring program source codes 214
Appendix III. File management source codes 231
Appendix IV. Video control routines 262
Appendix V. Linkage map files 273
Appendix VI. Command files to build system 302
Appendix VII. Functions for time and date calculations 321

LIST OF FIGURES

Number		Page
1	Overview of computer system use in study	3
2	Interconnection of CCI, scanner, data measurement system, signal conditioning interfaces, and transducers	9
3	Schematic diagram of temperature interface	16
4	Connection of temperature interface to data measurement system	17
5	Schematic diagram of conductivity interface	20
6	Connection of conductivity interface to data measurement system	21
7	Schematic diagram of voltage to frequency interface	22
8	Schematic diagram of pump control interface	24
9	Connection of pump control interface to data measurement system	25
10	Connection of flow measurement transducer to data measurement system	26
11	Connection of anemometer to data collection system	27
12	Foreground - background memory and interaction area	41

LIST OF TABLES

Number		Page
1	Equipment Comprising Computer System	4
2	Addresses and Vectors of Output Devices	4
3	Addresses and Jumper Arrangement for DLV11-E	6
4	Computer Calling Unit Switch Settings	7
5	Addresses and Switch Setting for the Common Carrier Interface	10
6	Addresses and Switch Setting of HP-IB Measurement System ASCII Interface	10
7	Format of Digital Output from ASCII Interface	11
8	Format of Digital Output from ASCII Interface as Used in Data Collection at Boulder	12
9	Switch Settings for 5302A MHz Universal Counter	13
10	Switch Settings of Scanner for Address ASCII ')'	13
11	Pin Assignments for the Interface Bus Backplane	15
12	Parts List for Temperature Interface	18
13	Parts List for Conductance Interface	19
14	Parts List for Frequency Interface	23
15	Parts List for Pump Control Interface	28
16	Bit Assignments for Digital-to-Analog Converter	28
17	SYCND.MAC - RT-11 V04.00B System Conditional File	31
18	System Configuration and System Options as Given by the SHOW Command	32
19	Programming Steps Functions Along with Octal Codes, Required Handshake Signal, and Diagnostic Values for Byte Location JBCTL	34

Number Page

20 HP-IB Programming Steps Included in the Control Subroutines 38

21 Channel Assignment for Scanner 39

22 Byte Locations for Program Interaction 42

23 Main Subroutine Data Assignments for JBDS1 42

24 Secondary Subroutine Data Assignments for JBDS2 43

25 Miscellaneous Data Assignments of Variables 43

INTRODUCTION

Rapid infiltration has been shown to be an effective method for the treatment of wastewater, but treatment is dependent on the frequency and duration of wastewater application. Improper application-drying cycles can cause a high concentration of a pollutant to reach groundwater at the beginning of an application period (Leach and Enfield 1983). Maintaining a constant water potential in the soil can eliminate this problem while maintaining a high population of microorganisms to biodegrade the waste pollutant. This type of system requires a high degree of control and attention (Enfield 1977). Automating the control of a rapid infiltration system allows for wastewater to be applied at the optimum moisture levels for soil organism populations which in turn promote transformation of biodegradable compounds.

Although manual control systems have been shown to be effective for the operation of rapid infiltration sites they cannot effectively maintain the moisture conditions at the optimum levels for treatment. Manual control systems have no provisions for continuously monitoring environmental conditions. Analog control systems can control pumps and relays, but cannot retain data and make complex comparisons and do not have the ability to easily change control parameters. A digital control system has the advantage of both operating a treatment system while storing operational data but at an increased cost. Interfacing digital computer software and hardware into a total system allows data collected at the site to be evaluated and to maintain moisture levels within a reasonable predetermined level. The hardware consists of the computer itself, the data collection system, the signal conditioning interfaces and the transducers and sensors measuring environmental parameters. All the hardware components except the computer were located in Boulder, Colorado. This study was an outgrowth of a manual control study whose results are presented in Volume I. The software consists of the control program, the monitoring program, the file management program and a telecommunications program for transferring the data to another computer for backup storage and statistical analysis.

CONCLUSIONS

1) Remote control of a land treatment site has been demonstrated to be feasible.

2) Field installed sensors had limitations in keeping environmental conditions at the optimum level for treatment of wastes. Reliable and accurate sensors were the main constraint in obtaining environmental and operational data.

HARDWARE COMPONENTS

The hardware components of the land treatment control system include a PDP 11/23 computer system, a data collection system, and signal conditioning interfaces. The integrated system allowed data to be collected at Boulder, Colorado and transmitted to Ada, Oklahoma, for computer analysis. If needed, codes were transmitted back to Boulder to turn the wastewater distribution system pumps on or off. Each of the main components of the hardware system are discussed individually.

COMPUTER SYSTEM

An overview of the computer system is shown in Figure 1. This system was integrated with a modem, an IEEE 488 bus data collection system and signal conditioning interfaces to collect data and control the Boulder land treatment system. Table 1 lists the components of the computer system. The PDP 11/23 is a 16 bit micro processor which supports memory management along with floating point processing [KEF11-AA] (Digital Equipment Corp. 1979a). Earlier versions of the PDP 11/23 supported 18 bit data addressing with a maximum of 256k bytes of memory. Newer versions support 22 bit addressing to a maximum of 4 Mbytes of memory. The CPU used in this project utilized 18 bit data addressing. The DLV11-J was used as the interface to the system console (VT100), the foreground job terminal (LA120), and a serial line printer (LA120). The LA120 terminal was switch selectable from printer to foreground terminal. One DLV11-E board was used to interface the remote job terminal and modem to the PDP 11/23. The other DLV11-E was used for the communication link to other computer systems. The expansion box (BA11-NE) allowed for the inclusion of the DLV11-E cards and also gave extra room for system expansion.

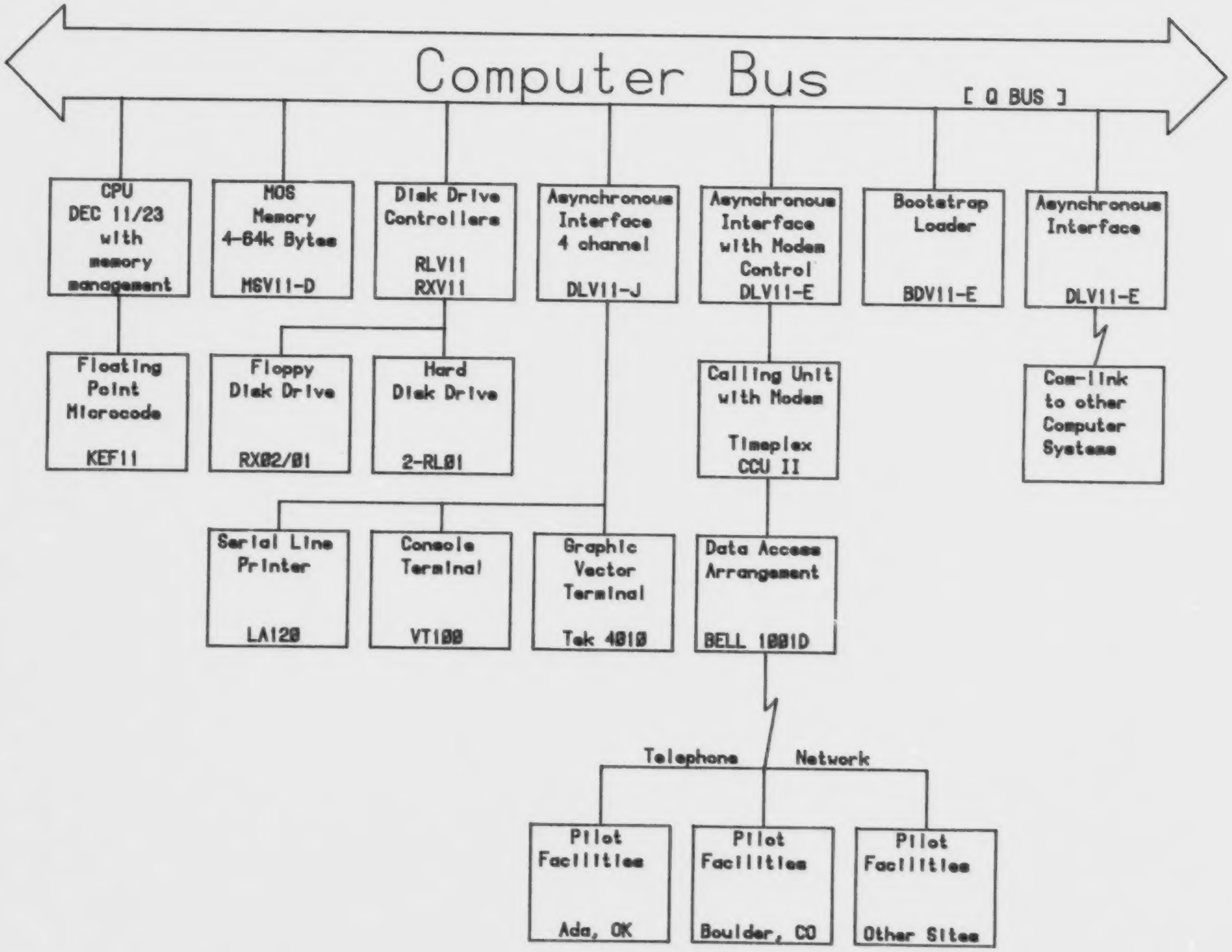

Computer Bus
[Q BUS]
CPU DEC 11/23 with memory management
MOS Memory 4-64k Bytes MSV11-D
Disk Drive Controllers RLV11 RXV11
Asynchronous Interface 4 channel DLV11-J
Asynchronous Interface with Modem Control DLV11-E
Bootstrap Loader BDV11-E
Asynchronous Interface DLV11-E
Floating Point Microcode KEF11
Floppy Disk Drive RX02/01
Hard Disk Drive 2-RL01
Calling Unit with Modem Timeplex CCU II
Com-link to other Computer Systems
Serial Line Printer LA120
Console Terminal VT100
Graphic Vector Terminal Tek 4010
Data Access Arrangement BELL 1001D
Telephone Network
Pilot Facilities Ada, OK
Pilot Facilities Boulder, CO
Pilot Facilities Other Sites

Figure 1. Overview of computer system used in study

TABLE 1. EQUIPMENT COMPRISING COMPUTER SYSTEM

SR-WXLLB-BK	PDP 11/23 RT-11 DUAL RL01 SYSTEM
system includes:	
DLV11-J	four asychronous serial interfaces
BDV11	bootstrap diagnostic terminator
MSV11-D	64Kb MOS memory boards (2)
VT100-AA	video terminal
Additional hardware from DEC:	
VT1XX-AB	VT100 Advanced video option
VT100-AC	VT100 Printer Port option
DLV11-E	Asynchronous line interface (2)
MSV11-DD	64 K MOS memory boards (2)
KEF11-AA	floating point microcode chip
BA11-NE	large expansion box
BCV1B-06	expansion cable assembly
LA120-DA	EIA DECWRITER III terminal
RL01K-DC	RL01 5.0 Mb disk cartridge (2)

Note: Identifying numbers and order codes are those at the time the system was ordered. These numbers may not be currently valid.

The addresses of the terminal output devices are given in Table 2. These addresses and vectors along with the baud rates were hardwired on each board. The standard addresses and vectors as specfified by Digital Equipment Corporation (DEC) were used for the other devices in the system (Digital Equipment Corporation 1981a).

TABLE 2. ADDRESSES AND VECTORS OF OUTPUT DEVICES

UNIT	TYPE	ADDRESS	VECTOR	MODULE	BAUD
TT0	CONSOLE	177560*	60,64*	DLV11-J	9600
LS	PRINTER	176500	300	DLV11-J	9600
TT1	LOCAL	176510	310	DLV11-J	9600
TT2	LOCAL	176520	320	DLV11-J	300
TT3	REMOTE	176530	330	DLV11-E	300
	COM-LINK	175640	340	DLV11-E	VARIED

* OCTAL NUMBER

The DLV11-E used for telecommunications to the remote field site was hardwired to give a baud rate of 300 with even parity and a 5 bit data type. This was required in order to communicate with the Hewlett-Packard data collection system and the IEEE 488 bus. The DLV11-E utilized in the communications link was wired to give a programmable baud rate. This way it could be used with 300/1200 modems and also directly connected at 9600 baud to a PDP 11/70 computer sytem. The wiring configuration for both DLV11-E are given in Table 3.

A cable from the remote DLV11-E card was connected to a computer calling unit (CCU) through the RS-232 data interface. The CCU was in turn connected to a BELL data arrangement through the coupler interface. The RS-366 and modem external interfaces were not used in this system. The CCU and a BELL data arragement comprise the link between the computer and the switched telephone lines. The CCU translates the digital data furnished by the computer system to dial pulses recognizable by the telephone network. The CCU included the following:

Timeplex CCU II computer calling unit
D-DC diagnostic card for manual dialing
BELL 103 compatible internal modem

See table 4 for the wiring configuration for the CCU.

TABLE 3. ADDRESSES AND JUMPER ARRANGEMENT FOR DLV11-E*

Jumper Designation	Remote DLV11-E	Com-link DLV11-E	Jumper Designation	Remote DLV11-E	Com-link DLV11-E
A3	I	R	BG	I	I
A4	I	R	P	I	R
A5	R	I	E	R	R
A6	I	R			
A7	R	I	1	I	R
A8	I	I	2	R	R
A9	R	I			
A10	I	R	PB	R	I
A11	I	I	C	I	I
A12	I	I	C1	I	I
			S	R	R
V3	I	R	S1	R	R
V4	I	R			
V5	R	I	H	R	R
V6	I	I	B	R	R
V7	I	I	-B	I	I
V8	R	R	FD	I	I
			RS	I	I
R0	R	R	FB	I	I
R1	I	I	EF	R	R
R2	R	R			
R3	I	I	MT	R	R
			M	R	R
T0	I	I	M1	R	R
T1	R	R			
T2	R	R			
T3	R	R			

R = Jumper removed
I = Jumper installed
* Digital Equipment Corporation 1981(a),
Digital Equipment Corporation 1977.

TABLE 4. COMPUTER CALLING UNIT SWITCH SETTINGS

D-CC board

CBT Program socket	all pins jumpered					
B-A	Installed					
X-X	Installed					
Y-Y	Installed					
ACR timer adjust	switch	1	2	3	4	
		on	off	off	on	(60 second)

all other jumpers and chips removed

D-DC Diagnostics board

C-6 (Dial bit decoder) ASCII characters

1 --> 6
2 --> 7
3 --> 10
4 --> 11

D-4 Character speed (300 baud)

A	Installed
B	Not installed
C	Installed
D	Installed
E	Installed
F	Not installed
G	Not installed
H	Not installed

H-6 ASCII parity

No strap EVEN Parity

Bell 103 compatible modem board

Transmitter signal
thumbwhell setting at -3 dBm

Program chips locations* of installed straps for CBT mode.

- P-1
  - E
- P-2
  - A, B, C, D, F, G
- P-3
  - A, B, C, D, E, G

All other straps removed

---

* Timplex 1977a, Timplex 1977b.

DATA COLLECTION SYSTEM HARDWARE

The data collection system took the signals generated from the signal conditioning interfaces and converted this signal to a serial input (RS-232) which was transmitted back to the computer system (Figure 2). The data collection system utilized an IEEE-488 bus for communication.

The data collection system consists of the following units connected on the Hewlett-Packard Interface Bus (HP-IB).

- Hewlett-Packard Common Carrier Interface Model 59403A
- Hewlett-Packard Data Measurement System which contains:
  - a) Data Measurement System 5300B
  - b) Hewlett-Packard ASCII Interface Model 5312A
  - c) Hewlett-Packard Universal Counter Model 5302A
- Hewlett-Packard Scanner Model 3495A

Each of the units was interconnected through software control to either measure a data point or to turn the sprinkler system on or off. Programming considerations of these units are discussed in the software section.

The Common Carrier Interface (CCI) is an interface between the serial asynchronous data port of the modem and the HP-IB. The CCI converts the data from bit parallel format to a bit serial format which can be read by a serial computer port. Hewlett-Packard recommends CCI units be operated in pairs, one as a local CCI at the computer and another CCI at the remote site to convert data received by the modem into a HP-IB format. The local unit and controller in our configuration was simulated by using software and a DEC DLV11-E asynchronous interface.

The front panel has indicator lights for five service request functions (parity error, handshake error, line disconnect, abandon call and retry, call complete), two mode indicators (handshake, online), sync detect indicator, and two address functions (listen and talk). If any of these functions are detected, the appropriate indicator lights are activated. There are also four sets of switches. They are power on/off, baud rate, interface, and parity. See Table 5 for settings of these external switches and the internal address switches. The internal switches are used to set the talk and listen addresses as well as to set the unit into a permanent handshake and/or on-line mode. The manual gives a full discussion of these modes and operation of this unit. (Hewlett-Packard, 1976).

The rear panel contains the power connector, a RS-232C serial input connection, a HP-IB bus interconnecting port, a dialer port, dedicated line port, and a logic probe connection. In this study, the RS-232 port was used to connect this unit to the modem, and the HP-IB port was the link to the HP-IB bus.

The CCI is a multi-purpose unit which can be operated in many modes and with many devices. A problem was encountered with the CCI when unclean electrical power, mainly caused by thunderstorms, caused the CCI to enter a hung state. To remedy this condition, the unit had to be manually cycled on and off. After installing a line conditioner to the AC input power, this problem was not encountered again.

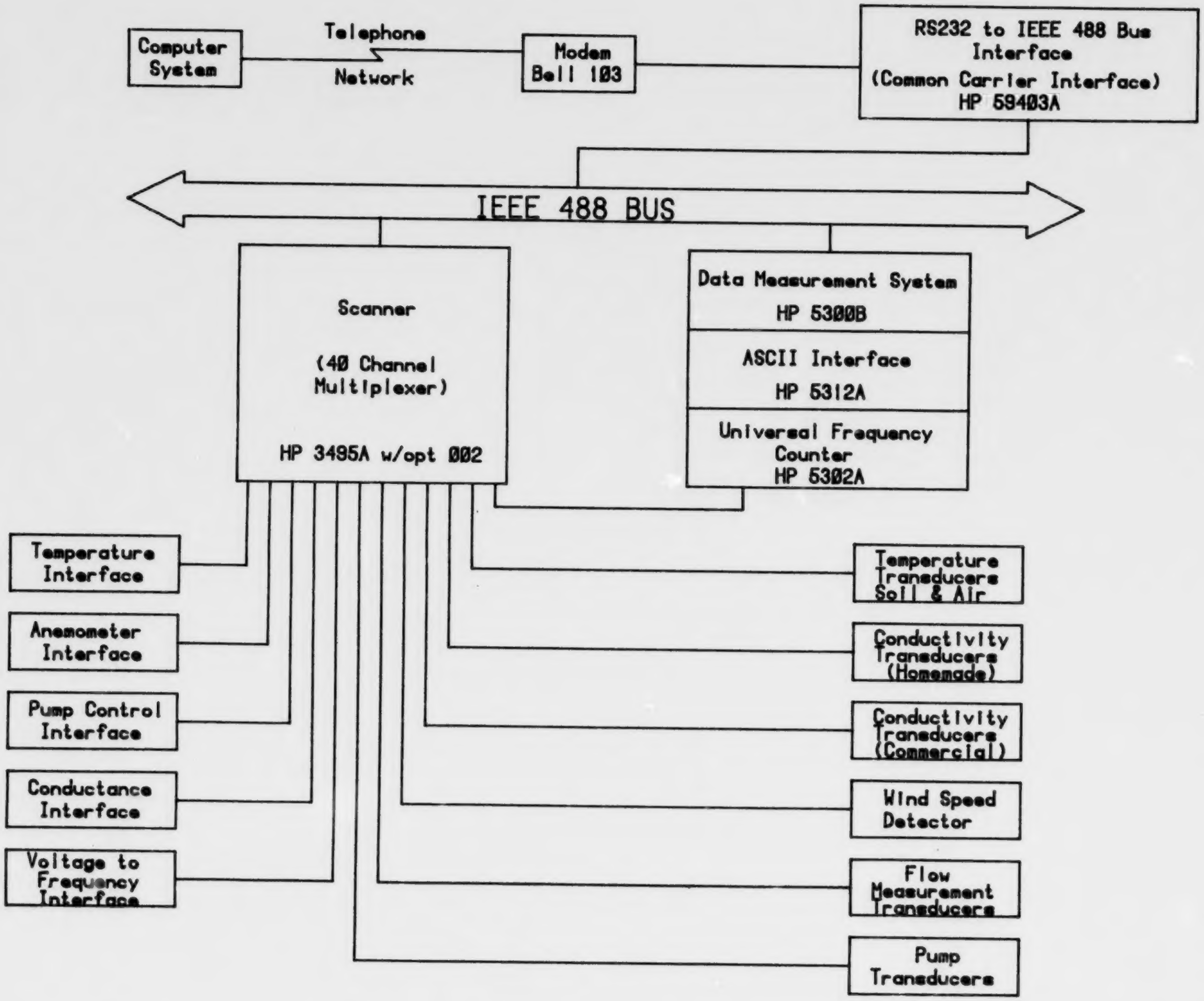


Figure 2. Interconnection of CCI, scanner, data measurement system, signal conditioning interfaces, and transducers

TABLE 5. ADDRESSES AND SWITCH SETTING FOR THE COMMON CARRIER INTERFACE

Internal switches switch number	status	comment
1	off	address of remote CCI ASCII 'R'
2	on	
3	on	
4	off	
5	on	
6	off	permanent on-line mode
7	off	permanent handshake mode
Front panel switches		
Baud Rate	300	
Interface	ASYN	
Parity	EVEN	

The measurement system mainframe (5300B) is a solid state measuring device which is plug connected to a variety of interconnecting modules to comprise a data sensing system (Hewlett-Packard 1977). The mainframe unit itself contains a LED display assembly, a variable sampling rate switch, and the power connector. Other modules are added to the mainframe to complete a total data sensing system as the mainframe itself can only display a given measurement.

The ASCII interface (5312A) is an inter-plug module for the measurement mainframe (Hewlett-Packard, 1975a). The module is between the mainframe and any other associated modules such as a counter. This module provides the interface to the HP-IB. This unit provides the measurement data format, which includes all the significant digits, exponents, overflows, error codes, and miscellaneous function codes to the bus. The front panel contains a talk and listen indicator. These indicators are illuminated when the unit is addressed to talk or listen, respectively. The rear panel contains the connector to the HP-IB and the addressable DIP switches. The DIP switches are used to program both the talk and listen addresses (Table 6). Table 7 shows the general format of the digital output to the interface bus after the interface has been addressed to talk. The particular format received will depend upon which type of counter unit is present. The format used in this project is given in Table 8.

TABLE 6. ADDRESSES AND SWITCH SETTING OF HP-IB MEASUREMENT SYSTEM ASCII INTERFACE

Self-check	Shield	Talk only	A5	A4	A3	A2	A0
1	0	1	0	1	0	1	0

ASCII address codes determined by switch setting A5 - A1.

Listen * 1 = switch setting to on.
Talk J 0 = switch setting to off.

TABLE 7. FORMAT OF DIGITAL OUTPUT FROM ASCII INTERFACE

Output order: Bytes		
1	0	Overflow or blank
2	F S V R	Annunciators (Hz,Seconds,Volts,Ohms,Blank)
3		Blank
4	D	Digit or blank
5	D	
6	D -	Digit, blank or minus sign
7	D .	Digit or decimal
8	D .	
9	D .	
10	D .	
11	D .	
12	D	Digit only
13	E	E for scientific notation
14	+ -	Plus or minus sign
15	0 3 6 9	Exponent
16	CR	Carriage return
17	LF	Linefeed

TABLE 8. FORMAT OF DIGITAL OUTPUT FROM ASCII INTERFACE USED IN DATA COLLECTION AT BOULDER

Output order: Bytes			
1	0		Overflow or blank
2	F		Annunciator (Hz)
3			Blank
4	D		Digit or blank
5	D		
6	D		Digit
7	.		Decimal
8	D		Digit
9	D		
10	D		
11	D		
12	D		Digit only
13	E		E for scientific notation
14	+	-	Plus or minus sign
15	0	3 6 9	Exponent
16	CR		Carriage return
17	LF		Linefeed

The universal counter (5320A) is plug connected to the ASCII interface which, in turn, was plug connected to the measurement mainframe. The counter is capable of measuring frequencies, time intervals, period and period averages, ratio of two frequencies, and totalizing. The actual type and range of output is determined by how the switches are set on the front panel. Switch settings are given in Table 9. A detailed description of the function and operating features are given in the operating and service manual. Data collected for this project used the frequency mode of output. One should pay particular attention to the discussion on setting the sensitivity of this unit in the equipment reference manual (Hewlett-Packard 1975b).

The data signals from the signal conditioning interfaces were connected to the appropriate signal input connectors on the front panel of this unit.

TABLE 9. SWITCH SETTINGS FOR 5302A MHz UNIVERSAL COUNTER

Sensitivity (left)	as per manual
Wave form (left)	middle
Time base	.1 us
Function	B
Wave form (right)	middle
Sensitivity (right)	as per manual
Sampling Rate	as per manual

The scanner is a system instrument which can be fitted with up to four programmable channel switching options (Hewlett-Packard 1974). Four Option 002, Accessory 44402A, 10 channel relay actuator assembly units were installed in the scanner main unit. Each actuator assembly relay consists of one decade or 10 channels. This gave the scanner a total of 4 decades or 40 available channels. Any or all channels can be opened or closed simultaneously. The internal switch settings are given in Table 10. This unit can only be addressed to listen. The HP-IB connector cable is located on the back panel.

TABLE 10. SWITCH SETTINGS OF SCANNER FOR SCANNER ADDRESS ASCII ')'

ADDRESS BIT	STATUS
7	OFF
6	ON
5	OFF
4	ON
3	OFF
2	ON
1	OFF

duodecade addressing as per manual*

*Hewlett-Packard 1974.

SIGNAL CONDITIONING INTERFACES

A variety of transducers are used in the control of the treatment system. Each of the transducers had individual characteristics. It was necessary to construct a series of interfaces which could translate the signals to a common base. Frequency was selected since it was not affected by cable length. The output signal from the flow meters was in frequency and these signals were read directly by the measurement system. The anemometer signal was buffered and averaged over a 10 sec. period and then connected to a voltage-to-frequency converter. All of the instruments were connected to a common bus structure which interconected the instruments. The instrument bus was a 44 pin structure with pin assignments as listed in Table 11.

TEMPERATURE INTERFACE

The temperature interface is shown schematically in Figure 3. It consists of a current source to supply a constant current to the National Semiconductor LM335 temperature transducer, a buffer amplifier, A1; and signal conditioning amplifier, A2; which adjusts the gain to 100mV/°C and the offset to 0 VDC at -50° C. The LM335 temperature transducer is specified to have a maximum uncalibrated temperature error of 9°C and a maximum temperature nonlinearity of 1.5°C. Table 12 gives the parts used in the construction of the interface. Pin numbers listed are for an instrument bus used in conjunction with the scanner/multiplexer to interconnect appropriate instruments and transducers. Figure 4 indicates how the temperature interface was connected to the data collection system.

The temperature transducers were calibrated at one temperature with the cable installed to correct for offset due to resistance of the wire. After calibration the maximum error specified by the manufacturer is 2°C for the transducer. Additional errors may be present due to thermal drift of the interface circuit and variability in the contact resistances in the Hewlett-Packard Scanner. A 5.46K ohm reference resistor, installed in the scanner was used to evaluate the stability and reproducability of the temperature measurement. The mean effective temperature, based on the resistance of the reference resistor, was 5.8°C with a standard deviation of 3.8°C. If the errors are due to thermal drift in the current source, then the reported error is greater than the true error since the forward voltage drop is not directly proportional to current supplied.

TABLE 11. PIN ASSIGNMENTS FOR THE INTERFACE BUS BACKPLANE

Pin No.	Function
1	120 VAC
2	no connection
3	120 VAC
4	no connection
5	+15 VDC
6	Instrument common (Ground)
7	-15 VDC
8	Current source output to temp. sensor from interface
9	Voltage output from temp interface
10	Input to voltage/freq interface
11	Output from voltage/freq interface
12	Current input from conductivity sensor
13	Programmable gain (X 0.01) in conductivity interface
14	Programmable gain (X 0.10) in conductivity interface
15	Programmable gain (X 1.00) in conductivity interface
16	Conductivity interface output
17	Oscillator output to conductivity sensor from conductivity interface
18	Programmable gain common in conductivity interface
19	Voltage input from anemometer
20	Anemometer interface output
21	no connection
22	no connection
A	no connection
B	no connection
C	no connection
D	no connection
E	Treatment plant pump control relay input from scanner
F	Plot 1 (north plot) flow control relay from scanner
G	Distribution pump control relay from scanner
J	+5 VDC
K	+12 VCD
L	Treatment plant pump control relay output - operates a zero voltage switch that controls the motor starter at the treatment plant
M	Distribution pump control relay output - operates a zero voltage switch that controls the motor starter of the plots in conjunction with a liquid level switch which would prevent starting the pump without the presence of water
N	Plot 1 flow control signal from Westinghouse controller
P	Plot 2 flow control signal from Westinghouse controller
R	24 VAC
S	24 VAC
T	Distribution pump control relay from scanner
U	no connection
V	no connection
W	no connection
X	Digital-to-analog converter output in pump control interface
Y	no connection
Z	no connection

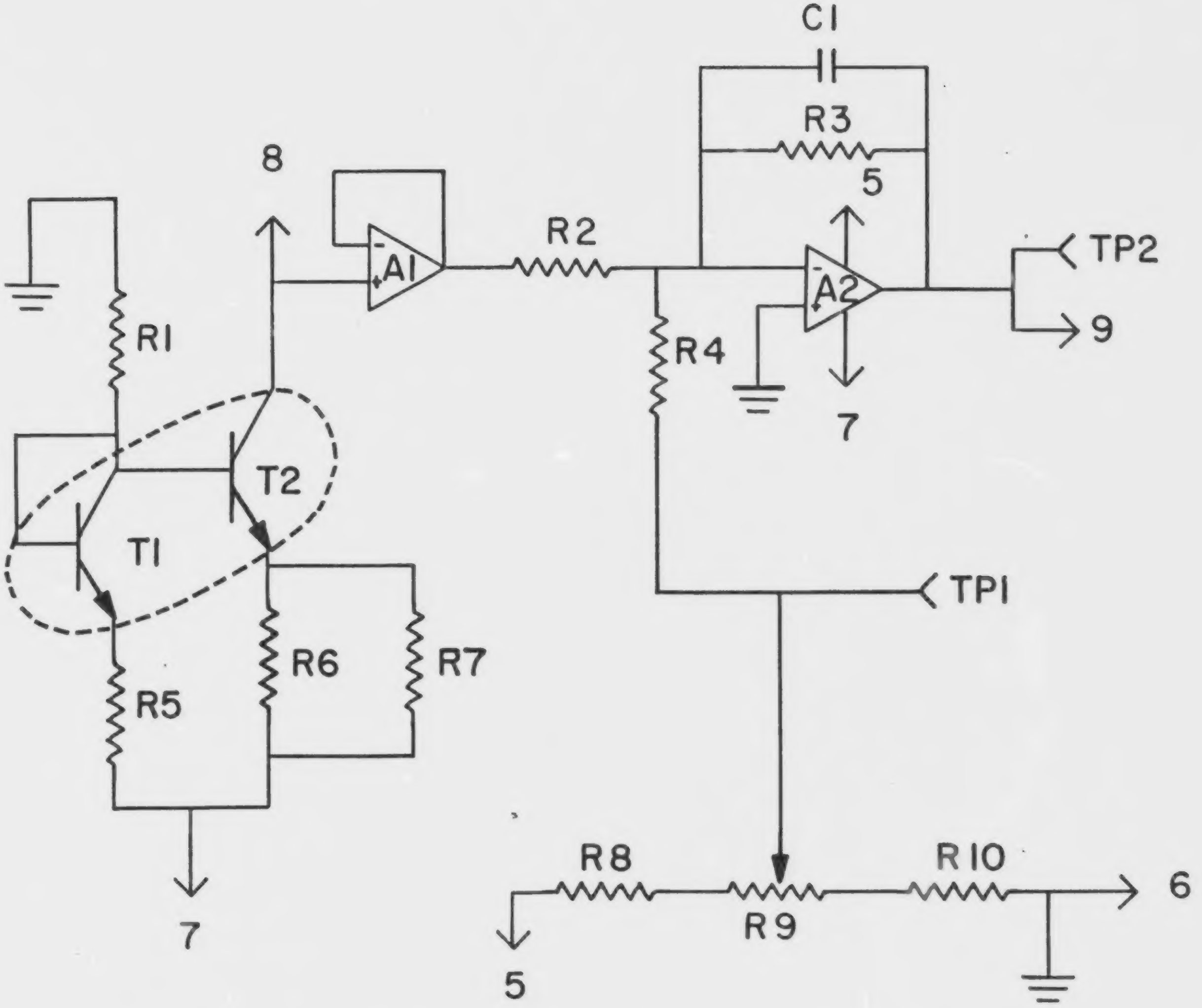


Figure 3. Schematic diagram of temperature interface

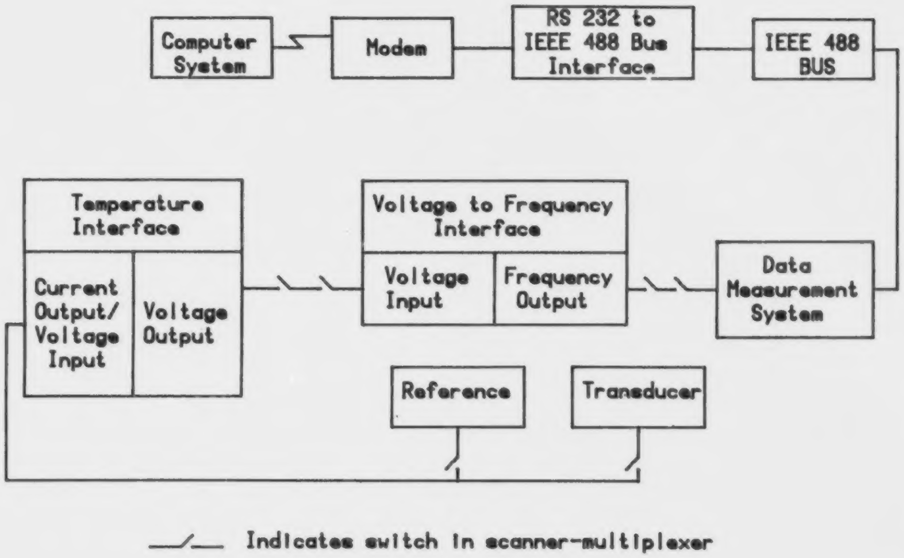


Figure 4. Connection of temperature interface to data measurement system

TABLE 12. PARTS LIST FOR TEMPERATURE INTERFACE

Part No.	Description	Comments
A1	Burr Brown 3501 OP Amp	
A2	Burr Brown 3501 OP Amp	
C1	0.22 μF	Polycarbonate
R1	15kΩ	1%
R2	10kΩ	1%
R3	100kΩ	1%
R4	10kΩ	1%
R5	1kΩ	1%
R6	2.7kΩ	1%
R7	selected for 0.5mA through pin 8	5%
R8	15kΩ	5%
R9	2kΩ	10%
R10	2.7kΩ	5%
T1	2N3904	heat sink to T2
T2	2N3904	

CONDUCTIVITY INTERFACE

The electrical conductivity was measured by applying an AC voltage to a conductivity transducer and measuring the current flowing after AC coupling the transducer to measurement circuitry. To interface with the remainder of the measurement system, the absolute value of the AC output voltage was obtained and filtered to yield a DC voltage proportional to the conductance. The interface contained programmable gain amplifier to facilitate differences in the positioning of the stainless steel rods used to assess the electrical conductivity of the soil water system. A circuit diagram of the conductivity interface is shown in Figure 5, and parts are listed in Table 13. Figure 6 indicates how the conductivity interface was connected to the data collection system.

The stability of the interface was determined using the frequency of the oscillator which maintained a mean frequency of 1.18 kHz with a standard deviation of .16 kHz. This variation in frequency should not measurably impact the results of the measurements, even though, theoretically the measurements are frequency dependent. When the reference resistor was installed in the conductance interface, a mean conductance of 2.5 mS was obtained with a standard deviation of .14 mS. The conductance of natural soil waters is temperature dependent with a temperature coefficient of approximately 2% per degree C. All of the conductance measurements were adjusted to reflect this temperature dependence. The deviation of the reference resistor then includes corrections due to variations in the reference temperature. The standard deviation reported in the reference conductance therefore includes errors in the reference temperature measurements. Variations in reference temperature account for the majority of the standard deviation in the reference conductance. The overall error in assessment of water content based on conductivity would be approximately 5%. This assumes the conductivity of the soil solution remains constant and is not influenced by differences in solution conductivity caused by rainfall events, variations in the conductivity of the wastewater applied or evapotranspiration.

TABLE 13. PARTS LIST FOR CONDUCTANCE INTERFACE

Part No.	Description	Comments
A1	Burr Brown 3501 OP Amp	
A2	Burr Brown 3501 OP Amp	
A3	Burr Brown 3501 OP Amp	
A4	Burr Brown 3501 OP Amp	
A5	Burr Brown 3501 OP Amp	
C1	0.26 μF	polycarbonate
C2	50 μF	electrolytic
C3	0.1 μF	polycarbonate
C4	2.2 μF	electrolytic
C5	2.2 μF	electrolytic
D1	1N4154	
D2	1N4154	
R1	10 kΩ	1%
R2	1 kΩ	1%
R3	100 kΩ	1%
R4	10 kΩ	
R5	10 kΩ	
R6	10 kΩ	
R7	54 kΩ	
R8	10 kΩ	
R9	1 kΩ	
R10	10 kΩ	
R11	10 kΩ	1%
R12	5 kΩ	1%
R13	10 kΩ	1%
R14	10 kΩ	1%
R15	selected for correct gain	
R16	40 kΩ	1%
R17	4.7 kΩ	
R18	2.2 kΩ	
R19	10 kΩ	1%

VOLTAGE-TO-FREQUENCY INTERFACE

The voltage-to-frequency interface converted the DC output voltages available from many of the transducers to a frequency which could be read by the Hewlett-Packard measurement system. All of the input signals were conditioned to have a range of zero to +10 VDC. The output of the interface was set such that the output would be 10 kHz/volt. The interface is shown schematically in Figure 7. Table 14 lists the parts for the interface. There was no independent measurement for stability of this interface. Most of the transducer interfaces utilized this interface (Figures 4, 6, 9, 11). Errors from this source are included in other measurements. The input acts as a summing junction for all signals applied. The double contacts at both the input and output of the voltage-to-frequency converter totally isolate the module from the remaining circuitry and prevent extraneous current inputs.

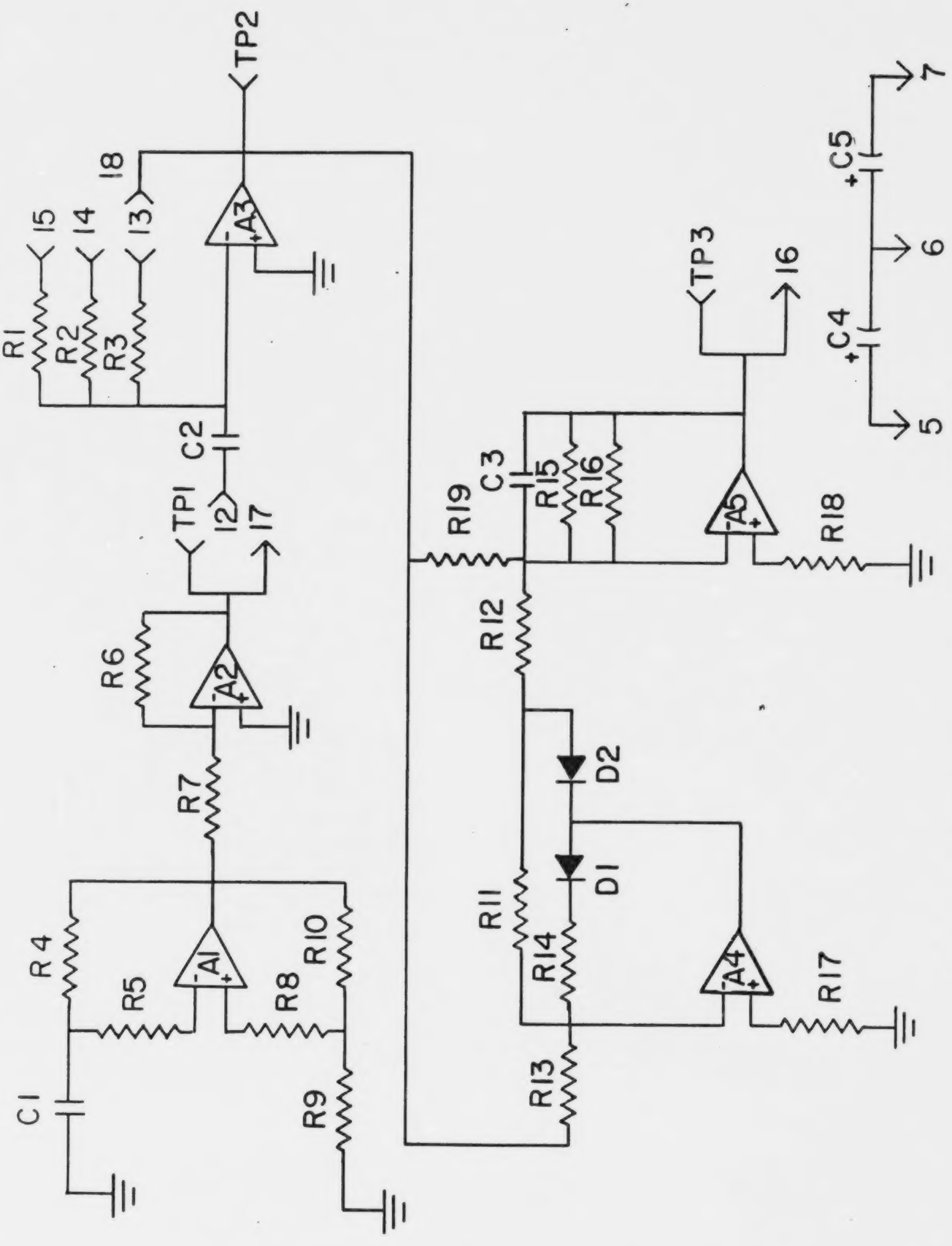


Figure 5. Schematic diagram of conductivity interface

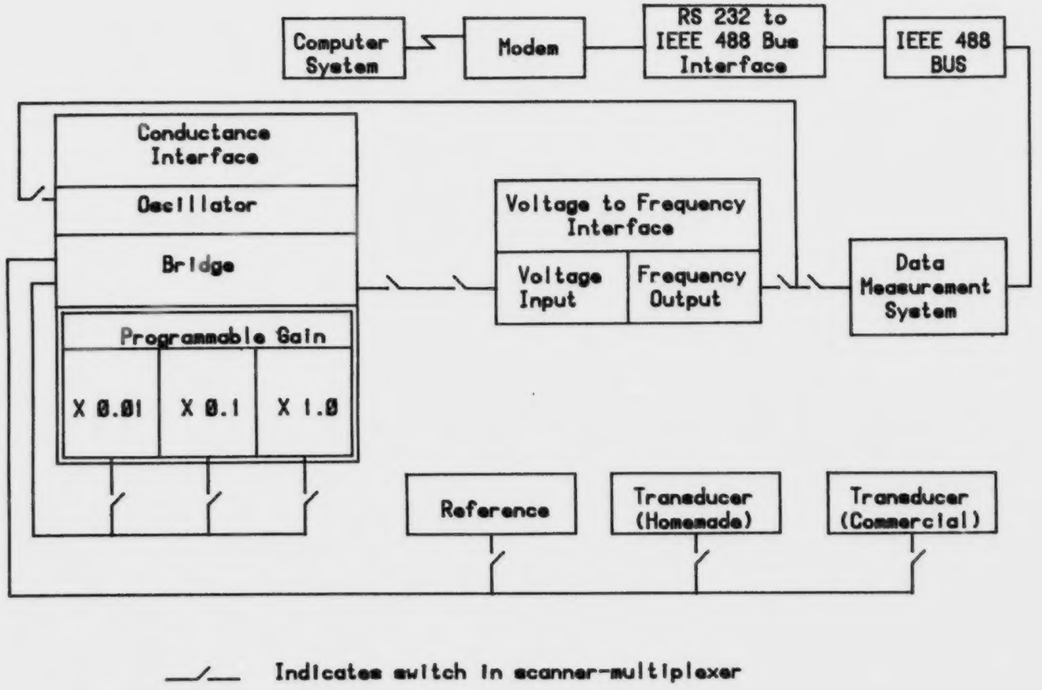
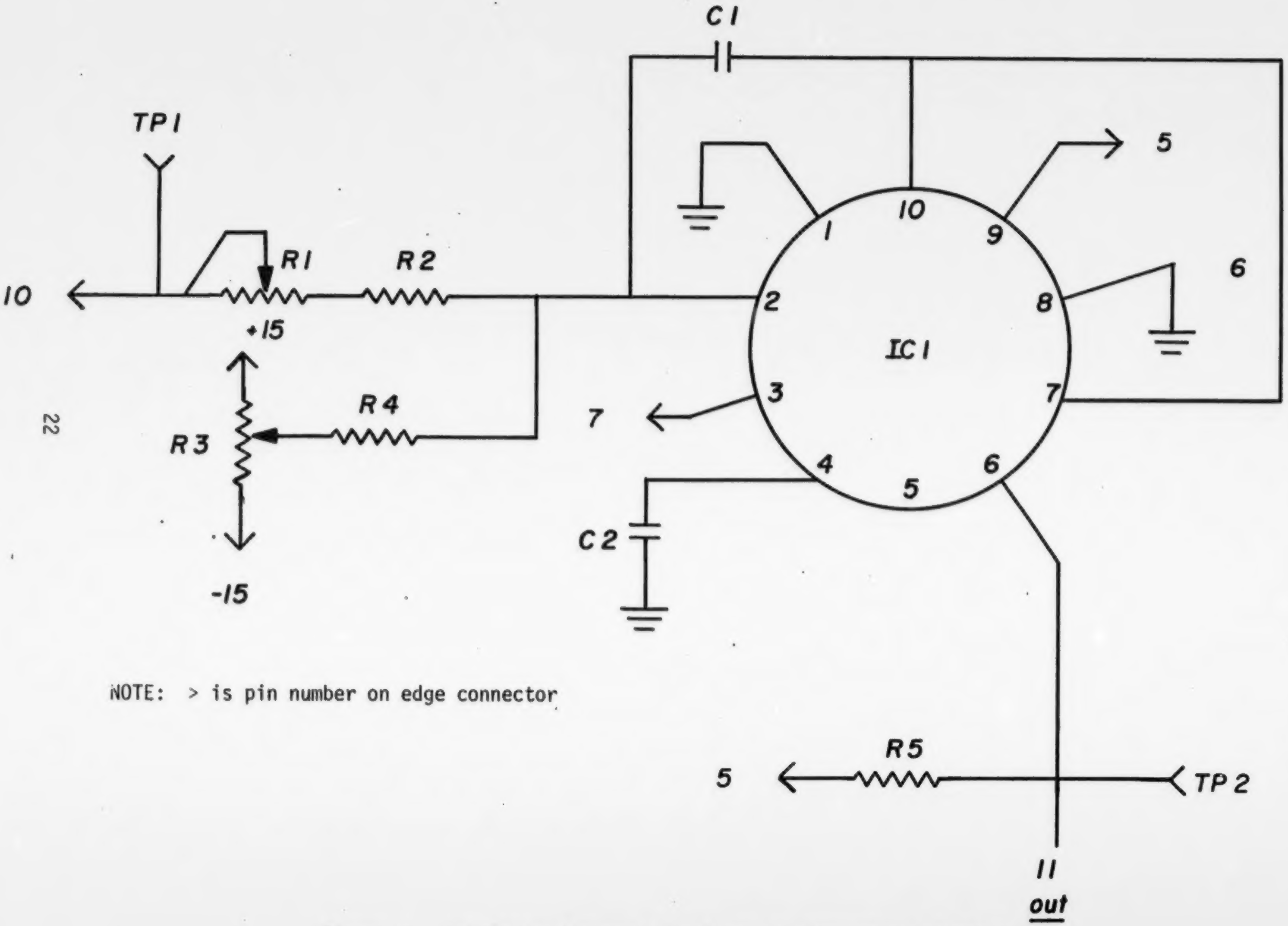


Figure 6. Connection of conductivity interface to data measurement system



NOTE: > is pin number on edge connector

Figure 7. Schematic diagram of voltage-to-frequency interface

TABLE 14. PARTS LIST FOR FREQUENCY INTERFACE

Part No.	Description	Comments
C1	0.001 μF	polycarbonate
C2	500 pF	polycarbonate
IC1	Burr Brown VFC32	Voltage-to-frequency converter
R1	10 kΩ	5%
R2	22 kΩ	1%
R3	50 kΩ	5%
R4	1 MΩ	1%
R5	1.8 kΩ	5%

PUMP CONTROL INTERFACE

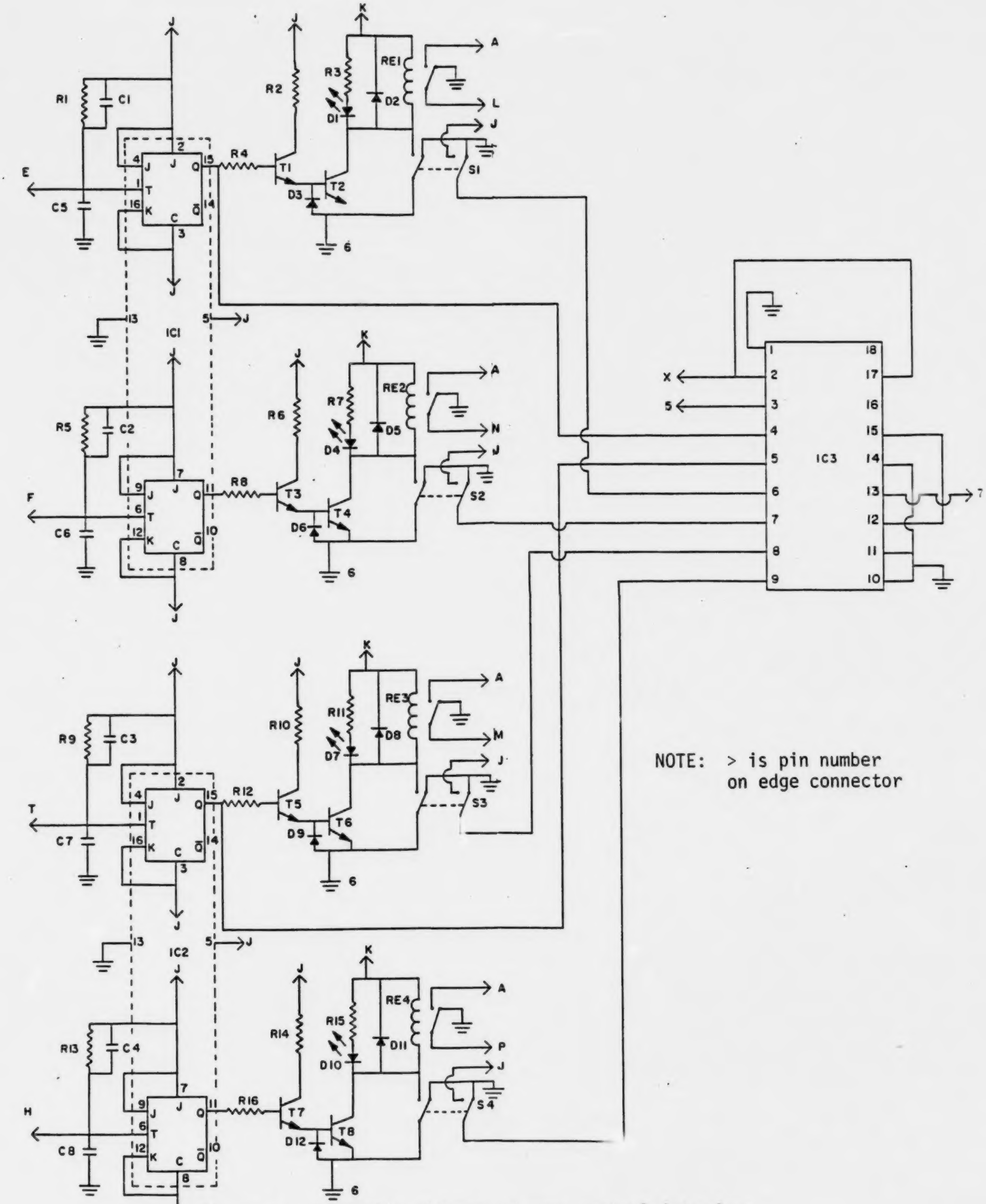
It was necessary to have the capability of operating the sprinkler system either manually at the treatment site or automatically through computer control. Simultaneously, the computer needed to know if the sprinkler was being operated manually. The Pump Control Interface provided the logic and relays to operate the pumps and valves as well being able to determine the status of these relays. The logic was developed using 7476 J-K flip flops in a divide by 2 mode. Each contact closure of the appropriate channel in the Hewlett-Packard Scanner would change the status of the flip flop. It was necessary to know the status of the flip flop because there was contact bounce in the Scanner relays. On occasion, this bounce would act as a double contact closure or no change in status. Six bits of an 8 bit Digital-to-Analog Converter were used to determine the status of the flip flops and manual switches. The flip flops controlling the plot valves were not monitored since the status could be indirectly monitored using the flow meter. The circuit diagram is shown in Figure 8 and parts listed in Table 15. Bit assignments are listed in Table 16. Figure 9 indicates how the pump control interface was connected to the data collection system.

FLOW MEASUREMENT

Flow measurements were read directly using a ALTOFLUX MT 900 Magnetic-Inductive flowmeter. This reading was read directly by the data measurement system (ALTOMETER). See Figure 10.

ANEMOMETER INTERFACE

The anemometer interface was connected directly to the voltage-to-frequency converter (Figure 11). The wind speed detector itself was placed on top of the equipment building so it would not be obstructed by surrounding equipment.



NOTE: > is pin number
on edge connector

Figure 8. Schematic diagram of pump control interface

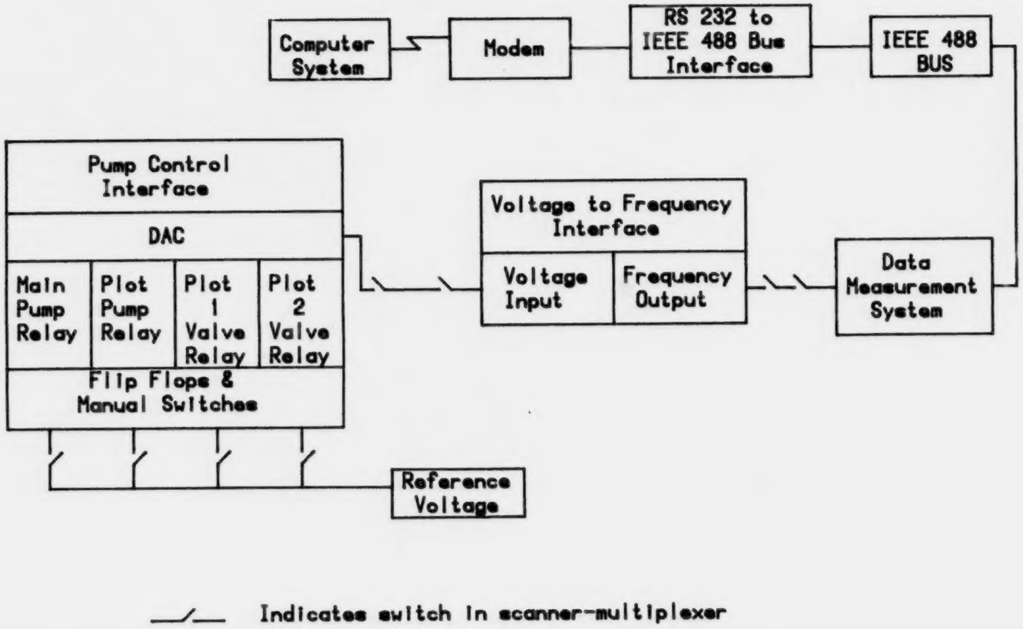


Figure 9. Connection of pump control interface to data measurement system

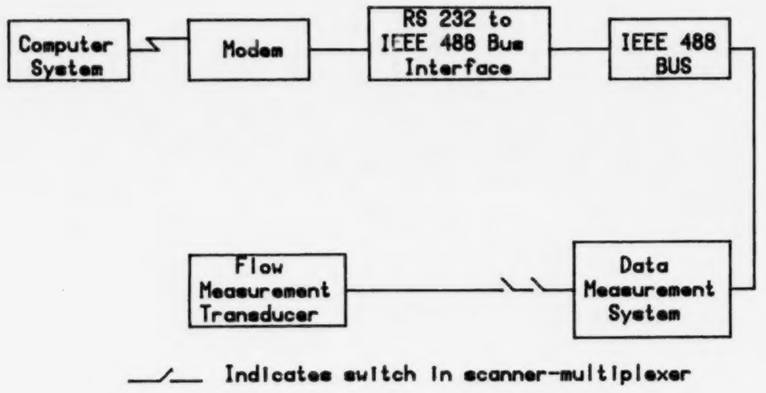

Computer System
Modem
RS 232 to IEEE 488 Bus Interface
IEEE 488 BUS
Flow Measurement Transducer
Data Measurement System
Indicates switch in scanner-multiplexer

Figure 10. Connection of flow measurement transducer to data measurement system

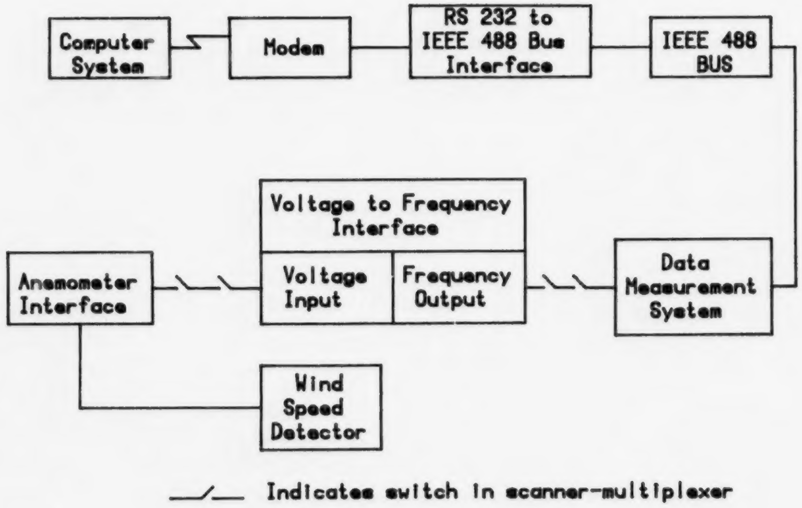


Figure 11. Connection of anemometer to data collection system

TABLE 15. PARTS LIST FOR PUMP CONTROL INTERFACE

Part No.	Description	Comments
C1-C4	0.1 μF	
C5-C8	0.22 μF	
D1		light emitting diode
D2-D3	1N4154	
D4		light emitting diode
D5-D6	1N4154	
D7		light emitting diode
D8-D9	1N4154	
D10		light emitting diode
D11-12	1N4154	
IC1-IC2	7476	J-K flip flop
IC3	DAC82KG	Burr Brown 8 bid Digital to Analog converter
R1	3.3 kΩ	5%
R2	4.7 kΩ	5%
R3	3.3 kΩ	5%
R4	100 kΩ	5%
R5	3.3 kΩ	5%
R6	4.7 kΩ	5%
R7	3.3 kΩ	5%
R8	100 kΩ	5%
R9	3.3 kΩ	5%
R10	4.7 kΩ	5%
R11	3.3 kΩ	5%
R12	100 kΩ	5%
R13	3.3 kΩ	5%
R14	4.7 kΩ	5%
R15	3.3 kΩ	5%
R16	100 kΩ	5%
RE1-RE4	SPDT 20A relay 12VDC coil	
S1-S4	DPDT switch	manual override on switch

TABLE 16. BIT ASSIGNMENTS FOR DIGITAL TO ANALOG CONVERTER

Pin	Bit	Assignment
4	1	Primary pump at clarifier flip flop
5	2	Sprinkler (plot) pump flip flop
6	3	Primary pump manual switch
7	4	Sprinkler (plot) pump manual switch
8	5	North plot (1 in computer program) manual switch
9	6	Center plot (2 in computer program) manual switch
10	7	Tied to ground (not used)
11	8	Tied to ground (not used)

COMPUTER SOFTWARE

INTRODUCTION

The software needed to control the land treatment system includes the distributed computer system software, specialty software for communications links to other computers, and developed software to actually control and monitor the site.

SYSTEM SOFTWARE

The operating system used was a system generated multi-terminal RT-11. RT-11 is a disk-based single-user real-time operating system which allows for dedicated online applications and interactive program development (Digital Equipment Corporation, 1981d). As this system is a single-user system, real-time turn around is very rapid. Batch facilities can be installed if desired.

There is a choice of three monitor programs with RT-11. They are single job (SJ) with the baseline monitor subset, foreground-background (FB), and extended memory (XM). These monitors are upward compatible, that is a program that will execute under the SJ environment will usually run under FB and XM monitors without modifications. The SJ monitor can run one job at a time, and its major advantages are its high data throughput and its low overhead leaving more memory for actual application programs. The FB monitor can execute two jobs concurrently, a foreground job and a background job. In addition to these, a system job may also be executing. An example of a system job is the error logger or printer queue. The foreground job is usually the real-time critical job and has priority over the other jobs in memory. If a foreground job does not have enough interrupts it can take total control over the system and exclude all other jobs. A job like this should be run under the SJ monitor. The XM monitor includes all the features of the FB monitor plus it allows programmed requests beyond the 64Kb address limit of the 16 bit CPU. These requests are made in assembly language and are not part of a higher-level language like FORTRAN.

RT-11 also has the capability of associating a logical device name with a physical system device. Output was written to logical unit BOU. BOU could be assigned to either one of the hard disks by the use of the ASSIGN command (Digital Equipment Corporation, 1981g).

NON-SYSTEM DISTRIBUTED SOFTWARE

In addition to the distributed software with RT-11 Version 4, the following sotfware from DEC was installed

QJ960-AQ	SSP-11 (scientific subroutine package)
QJ813-CQ	FORTRAN IV/RT-11 (version 02.5)

SSP was used for mathmatical and statistical analysis of the collected data. The actual control program was written in FORTRAN IV with some calls to the RT system library. The RT system library contains FORTRAN callable subroutines which give the programmer control over system memory and peripheral devices.

The following terminal emulator package from UNIQUE AUTOMATION PRODUCTS was also installed

402480	EZSHAR (EMULATOR SOFTWARE)

This package was used to transmit the data to other computer systems for data analysis and storage.

CONFIGURED OR SYSTEM GENERATED SYSTEM

The RT-11 FB system used in this project was generated and configured for specific hardware devices. The distributed FB system did not support multi-terminal devices and certain time-out features which were deemed to be necessary for adequate control of the land treatment site. A modified FB monitor was chosen for the following reasons:

1. The data collection program was to execute concurrently with a background monitoring program. This monitoring program sensed whether the program was executing properly or whether a piece of hardware caused the computer or data bus to hang.

2. The Timplex dialer and graphics output device required their own terminal support features. The Timplex dialer required that the telephone number be entered as an ASCII string for the phone to properly be dialed. The graphics terminal was used as an output device and not as an interactive terminal. This allowed graphics output to be entered on another terminal before it was displayed.

3. The total control system required two terminals, one for the foreground program output and one for the background program output. The BG program gives a constant update of the status of the FG program and, therefore, needs a dedicated output device. It was also found that the FG-BG monitor was more efficient if FG and BG jobs did not share the same terminal.

System generation is accomplished by answering hardware and system option questions using an interactive program supplied by DEC (Digital Equipment Corporation, 1981c). This program produces a system conditional file (Table 17) which is compiled and linked with the hardware and system monitoring software which produces a customized operating system for the specified hardware. Table 18 lists the operating system features resulting from the system generation process.

SOFTWARE CONTROL OF THE COMPUTER CALLING UNIT

To initiate control of the CCU, the data terminal ready (DTR) is set low and then set high. Setting the DTR low causes the phone to disconnect and clears the way for dialing a new number. When the DTR is set high, a phone number can be passed to the CCU. The number is an ASCII string prefaced by the number 1. This causes the CCU to go off hook and dial the number. At this point, the program would go into a loop to check for carrier detect. When the carrier was detected, control would pass back to the main program. Data can now be transmitted over the phone lines to the data collection system at Boulder. If the carrier was not detected after a set time, the dialing operation would be restarted. See BOUDAL for Appendix I.

TABLE 17. SYCND.MAC - RT-11 V04.00B SYSTEM CONDITIONAL FILE

```
SYSG$N   = 1            ;INDICATE SYSGENED MONITOR
TIME$R   = 1            ;ADD TIMER SUPPORT
RDF$L    = 1            ;ADD SYSTEM I/O ERROR MESSAGE
SYT$K    = 1            ;SYSTEM JOB SUPPORT
MTT$Y    = 1            ;MULTI-TERMINAL SUPPORT
MAT$S    = 1            ;ASYNCHRONOUS TERMINAL STATUS
MTI$M    = 1            ;TERMINAL TIME OUT
TIM$IT   = 1            ;DEVICE TIME OUT (REQUIRED BY TERMINAL TIME-OUT)
TTYOUT   = 40.          ;SIZE OF OUTPUT RING BUFFERS
TTYIN    = 132.         ;SIZE OF INPUT RING BUFFERS
ROL$OV   = 1            ;MONTH ROLLOVER SUPPORT
U$TIL    = 1            ;UTILITY COMMANDS
L$ANG    = 1            ;LANGUAGE COMMANDS
M$INI    = 1            ;MINIMAL COMMANDS
CLOCK    = 60.          ;POWER LINE FREQUENCY
STAR$T   = 1            ;STARTUP COMMAND FILE
FPU$11   = 1            ;FLOATING POINT SUPPORT
DY$CSR   = 177170       ;STATUS REGISTER FOR FIRST RX02
DY$VEC   = 264          ;VECTOR FOR FIRST RX02
DL$UN    = 2.           ;NUMBER OF RL01/RL02 UNITS
LS$CSR   = 176500       ;CSR FOR SERIAL LINE PRINTER
LS$VEC   = 300          ;VECTOR FOR SERIAL LINE PRINTER
DL11$L   = 4.           ;NUMBER OF LOCAL DL11 INTERFACES
DL11$M   = 1.           ;NUMBER OF REMOTE DL11-E LINES
DL11$N   = 3. + 1.      ;TOTAL NUMBER OF DL11 LINES
DLC$0    = 177560       ;CSR OF FIRST (CONSOLE) DL11
DLV$0    = 60           ;VECTOR OF FIRST (CONSOLE) DL11
DLC$1    = 176510       ;CSR OF SECOND (LOCAL) DL11
DLV$1    = 310          ;VECTOR OF SECOND (LOCAL) DL11
DLC$2    = 176520       ;CSR OF THIRD (LOCAL) DL11
DLV$2    = 320          ;VECTOR OF THIRD (LOCAL) DL11
DLC$4    = 176530       ;CSR OF FOURTH (REMOTE) DL11
DLV$4    = 330          ;VECTOR OF FOURTH (REMOTE) DL11
DZ11$N   = 0            ;NO DZ11 OR DZV11 SUPPORT
```

TABLE 18. SYSTEM CONFIGURATION AND SYSTEM OPTIONS AS GIVEN
BY THE "SHOW" COMMAND

```
RT-11FB (S) V04.00I

Booted from DL0:RT11FB
Resident Monitor base is 127236 (44702.)
USR is set SWAP
TT is set NOQUIET
Indirect file abort level is ERROR
Indirect file nesting depth is 3

PDP 11/23 Processor
FP11 Hardware Floating Point Unit
Extended Instruction Set (EIS)
KT11 Memory Management Unit
60 Cycle System Clock

Device I/O time-out support
Multi-terminal support
System job support

Device      Status       Vector
-------------------------------
  NL      Installed      000
  LS      Installed      300 304
  DY      Installed      264
  DL      Resident       160

TT  (Resident)
DL  (Resident)
    DL0 = SY , DK , BOU, INP
    DL1 = OUP
MQ  (Resident)
DY
LS   = LP
NL
3 free slots

      Unit Owner     Type       WIDTH   TAB   CRLF   FORM   SCOPE   SPEED

          0       S-Console DL   132     No    Yes    Yes    Yes     N/A
          1       Local     DL   132     No    Yes    Yes     No     N/A
          2       Local     DL    96     No    Yes     No     No     N/A
          3       Remote    DL    80     No    Yes     No     No     N/A
```

SOFTWARE CONTROL OF THE HP DATA COLLECTION SYSTEM

Control signals which come through the modem data line interface are then sent to the CCI and passed along the HP-IB. The CCI acts as the manager to the HP-IB. By sending the proper signals from the computer system one can cause the scanner channels to open or close, activate the data measurement system or multiplex them together to get a data reading.

The HP-IB bus has two modes of operation: The data mode and the command mode. While in the command mode, the CCI recognizes certain codes to be command functions. But in the data mode, the HP-IB must be addressed to listen for commands to be acted upon.

For the following sequences to work properly, the internal and external switches of the units must be set as discussed in the Hardware Section. This operation assumes asynchronous communication with the handshake and on-line modes activated. Certain commands require a handshake value to be returned before any further communications will take place. If the proper handshake value is not received the command can be resent until it is received. Table 19 gives the octal codes and required handshake characters. If after a few tries the proper handshake is not received, the bus needs to be cleared, and all the command sequences to that point retransmitted.

1. Abort remote bus (interface clear)

An interface clear signal is sent to the remote bus causing all data transfers to be aborted and the bus to be set to an unlisten or clear mode. This stops all data transfers on the bus. A hung bus can result if the proper handshake response is not returned.

2. Put bus in remote control

Informs the remote CCI that the next sequence is a command sequence and is to be interpreted as such.

3. Set bus in command mode

The command mode of the remote bus is activated.

4. Set bus to unlisten

When in the unlisten mode the CCI passes commands to the other units on the bus. Commands are now transparent to the CCI. Other units on the bus are set to an unlisten state and can only communicate on the bus if addressed to listen.

5. Talk address controller

This is the address of the controlling CCI. In multiple control systems, there can be more than one local-remote CCI pair. This address informs the bus which local is communicating. For our application, this was a dummy value which the bus was expecting, since there is no real local CCI. (See discussion on data system collection hardware.)

TABLE 19. PROGRAMMING STEPS FUNCTIONS ALONG WITH OCTAL CODES, REQUIRED HANDSHAKE SIGNAL AND DIAGNOSTIC VALUES FOR BYTE LOCATION JBCTL

STEP	FUNCTION	ASCII	OCTAL CODE	HANDSHAKE ASCII	HANDSHAKE OCTAL	JBCTL* VALUE
SETUP SCANNER						
1.	Abort remote bus (interface clear)	DLE	020			18
		C	103	F	106	19
2.	Put bus in remote control	DLE	020			20
		R	122			21
3.	Set bus in command mode	DLE	020			22
		A	101	F	106	23
4.	Set bus to unlisten	?	077	?	077	24
5.	Talk address controller	}	175	}	175	25
6.	Listen address scanner	)	051	)	051	26
7.	Set bus in data mode	DLE	020			27
		B	102			28
8.	Clear mode command	C	103	C	103	29
SEND CHANNEL NUMBERS TO CLOSE						
9.	Send scanner channel(s) to be closed.	FIRST DIGIT	060-067		FIRST DIGIT	060-067
		SECOND DIGIT	060-067		SECOND DIGIT	060-067
CLOSE CHANNELS						
10.	Execute commands to scanner	CR	015	CR	015	30

(continued)

*JBCTL refers to a program internal diagnostic value (See Table 22 and section on Foreground-Background Interaction.

TABLE 19. (Continued)

STEP	FUNCTION	ASCII	OCTAL CODE	HANDSHAKE ASCII	HANDSHAKE OCTAL	JBCTL* VALUE
	ACTIVATE DATA MEASUREMENT SYSTEM					
11.	Put bus back in remote control	DLE	020			1
		R	122			2
12.	Set bus in command mode	DLE	020			3
		A	101	F	106	4
13.	Set bus to Unlisten	?	077	?	077	5
14.	Send talk address of controller	}	175	}	175	6
15.	Listen address of counter	*	052	*	052	7
16.	Set bus in data mode	DLE	020			8
		B	102			9
17.	Initiate data collection system	I	111	I	111	10
18.	Set bus in command mode	DLE	020			11
19.	Set bus to unlisten	?	077	?	077	13
20.	Send talk address of counter	J	112	J	112	14
21.	Listen address of controller	}	175	}	175	15
22.	Set bus back in data mode	DLE	020			16
		B	102			17
	RETRIEVE DATA					
23.	Retrieve data from the data measurement system	LAST CHARACTER		NEXT CHARACTER		

*JBCTL refers to a program internal diagnostic value (See Table 22 and section on Foreground-Background Interaction.

6. Listen address scanner

Addresses the scanner and programs it to enter a listen mode. The scanner can now be sent commands.

7. Set bus in data mode

The bus is set in the data mode, so commands and channel numbers can be sent to the scanner.

8. Clear mode command

This causes the bus to recognize only management signals and lets other signals pass through the bus. This command also opens all scanner channels that were previously closed.

9. Send scanner channel(s) to be closed.

The scanner channels to be closed are now sent along the bus. To address an individual channel the decade of the channel is first addressed and then the unit of the channel is addressed. The format is to send the channel in two individual octal codes. That is, to close channel 8, one would send an octal 0 and then an octal 10, where 0 would be the decade and 10 (decimal 8) would be the unit number. Single digit channel numbers must be prefixed by a zero. To close channel 23 one would send an octal 2 and then an octal 3. Each digit must be echoed back before the next digit is sent. This handshaking ensures that the proper channels will be closed as desired. More than one channel can be closed at one time. If one needed to close channels 11, 25, 37; one would send the octal equivalents of these channels. The order the channels are sent is the order in which they are closed. This order can be random if desired. This channel closing ability allows a sensor to be chained to a signal conditioning interface, the voltage-to-frequency converter, and any other equipment desired.

10. Execute commands to scanner

After the channels to be closed are sent to the scanner, they need to be closed. This is accomplished by sending a carriage return character (CR) to the bus which tells the scanner to execute the previous commands. After this character is received the channels will then be closed.

11. Put bus back in remote control

Prepares remote bus to be put back into a command mode.

12. Set bus in command mode

The bus is now put back into a command mode so control characters can be recognized as commands and not data.

13. Set bus to unlisten

Make sure all the devices are in a nonlistening mode and not addressed.

14. Send talk address of controller

The dummy address of the local CCI was sent to satisfy the requirements of the bus.

15. Listen address of counter

Address the ASCII interface of the data measurement system to let it know that next sequence of commands are for it.

16. Set bus in data mode

The bus is now set in the data mode so data can be transmitted along it.

17. Initiate data collection system

The initiate command tells the data measurement system to take a reading and save this reading until it can be sent back along the bus to the computer.

18. Set bus in command mode

The bus is now put back into a command mode so another device can be addressed.

19. Set bus to unlisten

Make sure all the devices are in a nonlistening mode.

20. Send talk address of counter

Address the data measurement system (counter). This informs this device that it is to perform the instructions which follow.

21. Listen address of controller

This informs the bus that the data is to be sent to the controlling device, in this case the computer system (local CCI).

22. Set bus back in data mode

Sets the bus to data mode so that the next information to come down the bus is data and will not be recognized as a command sequence by any of the devices.

23. Retreive data from the data measurement system

The data are sent to the controller a byte at a time. To initiate transfer of the data, the last handshake signal which was present on the bus is sent to the measurement system. The measurement system then echoes back the first value in the talk output format. See Table 8. This process is repeated until all 17 bytes are sent. The correctness of the data format can be verified by checking the received data bytes with the talk data format. For example, bytes 16 and 17 are supposed to be a carriage return (CR) and a line-feed, respectively (LF). If something else is in these locations, the data bytes returned maybe suspect. To insure the correctness of the data format, the following data bytes were checked for the appropriate values: bytes 2, 7, 13, 16, 17. If any of these locations varied from the expected value, the data bytes were retransmitted. After the data bytes were collected, the octal codes were converted to real numbers of the magnitude which would correspond to ther location in the talk output format (Table 8). These values were then summed to give a data reading. (See BOUWRT for Appendix I.)

To effectively utilize computer resources, each programming step listed above was contained in a control subroutine. The FORTRAN code of the subroutines are given in Appendix I. The control subroutines and the programming steps they contain are given in Table 20.

TABLE 20. HP-IB PROGRAMMING STEPS INCLUDED IN THE CONTROL SUBROUTINES

SUBROUTINE	PROGRAMMING STEPS
BOURLL	1 TO 8
BOUCHN	9
BOURET	10
BOURTL	11 TO 22
BOUWRT	23

SAMPLING DATA AT REMOTE SITE

The channel assignments for the scanner are given in Table 21. These channels were multiplexed through programming the HP-IB so transducers and detectors could be connected to the data measurement system and the proper signal conditioning interface. The steps involved in reading a sensor were as follows:

1. Set up scanner.
2. Send channel codes to the scanner to multiplex sensors and signal conditioning interface to the data measurement system .
3. Close channels.
4. Activate data measurement system.
5. Retrieve data from site.
6. Verify data were in the proper range; if not, start sequence again. If after 3 tries data are still out of range, log the problem on the job terminal and proceed.
7. Store data on disk.
8. Start sequence for next sensor.

If a control code for the HP-IB did not respond with the proper handshake signal; the code was resent and the handshake rechecked. If after a few times, the correct code was not received, the subroutine that was in control (Table 20) would exit and be restarted. In some cases the whole sequence would be restarted from the beginning with step 1 above. BOUPRS.FOR (Appendix I) is an example of a routine to collect data from the remote site.

TABLE 21. CHANNEL ASSIGNMENT FOR SCANNER

Variable	Channel	Comment
IBCONB	00	Conductance bridge
IBCOO	01	Conductance oscillator output
MBCONR(1)	02	Cond Range x 0.01
MBCONR(2)	03	Cond Range x 0.1
MBCONR(3)	04	Cond Range x 1.0
IBFCI	05	Frequency counter input
IBVFI	06	Voltage/frequency input
IBVFO	07	Voltage/frequency output
IBTCM	08	Temperature conversion module
IBREFR	09	Reference resistor 5.46k
MBTMP(1)	10	Temperature Plot 1 station 1 3 cm
MBCON(1)	11	Conductivity Plot 1 station 1
MBPRS(1)	12	Comm Cond. plot 1 station 1
MBTMP(2)	13	Temperature Plot 1 station 2 3 cm
MBCON(2)	14	Conductivity Plot 1 station 2
MBPRS(2)	15	Comm Cond. plot 1 station 2
MBTMP(3)	16	Temperature Plot 1 station 3 3 cm
MBTMP(4)	17	Temperature Plot 1 station 3 30 cm
MBCON(3)	18	Conductivity Plot 1 station 3
MBPRS(3)	19	Comm Cond. plot 1 station 3
MBTMP(5)	20	Temperature Plot 2 station 1 3 cm
MBCON(4)	21	Conductivity Plot 2 station 1
MBPRS(4)	22	Comm Cond. plot 2 station 1
MBTMP(6)	23	Temperature Plot 2 station 2 3 cm
MBCON(5)	24	Conductivity Plot 2 station 2
MBPRS(5)	25	Comm Cond. plot 2 station 2
MBTMP(7)	26	Temperature Plot 2 station 3 3 cm
MBTMP(8)	27	Temperature Plot 2 station 3 30 cm
MBCON(6)	28	Conductivity Plot 2 station 3
MBPRS(6)	29	Comm Cond. plot 2 station 3
MBTMP(6)	30	Ambient Air temperature
IBWS	31	Wind speed
IBMPR	32	Main pump relay (normally open drain)
IBPPR	33	Plot pump relay (delay time before turning on)
MBFFT(1)	34	Plot 1 flow freq. transducer (determines flow rate)
MBFFT(2)	35	Plot 2 flow freq. transducer (determines flow rate)
MBPMP(1)	36	Plot 1 valve current relay (turns on valve controller)
MBPMP(2)	37	Plot 2 valve current relay (turns on valve controller)
	38	Reference resistor for temp check (same as 09 but used to eliminate crosstalk)
IBPNIC	39	flow test DAC converter

FOREGROUND - BACKGROUND INTERACTION

A unique aspect of this project was the communication link between the foreground and background programs. The DEC RT system library supports FORTRAN callable routines which can send data between the foreground (FG) and background (BG) jobs, but to utilize these routines both the FG and BG jobs have to be running concurrently and have synchronized input/-output processes (Digital Equipment Corporation, 1981e). If they are not synchronized, the FG job will pause and wait for the background jobs input/output requests before data will be transmitted again. This restriction made the use of the library routines unsuitable for our needs. A system was needed where the foreground job would execute independently of the background job.

A system was utilized where the FG job wrote its status to a particular byte addresses in the computer memory. The background monitoring job, when present, read the value in the appropriate address, converted the reading from a specific byte code to an memonic string which was displayed on the BG terminal screen. The BG program also could send information to the FG program in this manner (Figure 12).

The byte locations used were from locations 170 to 178. These are located in the trap and interrupt vector area part of the memory and are reserved by DEC for individual users (Digital Equipment Corporation, 1981b). This section of memory is not changed by normal FG/BG program execution which makes it an excellent location for storing numbers and data to be used in the interaction of the FG/BG jobs.

The bytes locations and functions are listed in Table 22. Associated with each location is a variable name which is used in the FG and BG programs to refer to that particular location. The location addresses are given in octal.

The input/output device can be set by using location JBDIO. This allows input and/or output to be from the FG terminal or the default disk file. Location JBDPGM informs the BG program whether the FG program is in a wait state before next sampling period or if the program had some type of error and terminated abnormally. Location JBDS1 is the name of the main subroutine that the program is currently executing (Table 23). Each subroutine had only one code except the routine to dial the telephone. This routine had a code for each critical operation of the dialing sequence (Table 23). As the program executes, each subroutine toggles the appropriate code in the interaction area. Location JBDS2 acts the same as the main subroutine locator, except the name of the secondary or control subroutine is sent (Table 24). The last channel number sent to the scanner is located in location JBCH1. JBCH2 is the location of the override byte. The override byte is set from the BG program. JBCTL is the last location in the control subroutines which was executed (Table 19). Handshake and some timing problems are easily diagnosed knowing the last control location. JBERR is the error byte. Critical equipment such as the conductivity bridge have an error code associated with them if an error occurs. These error codes are additive so that one can extract the exact error by knowing the total error code (Table 25). Other miscellaneous byte values are also listed in Table 25.

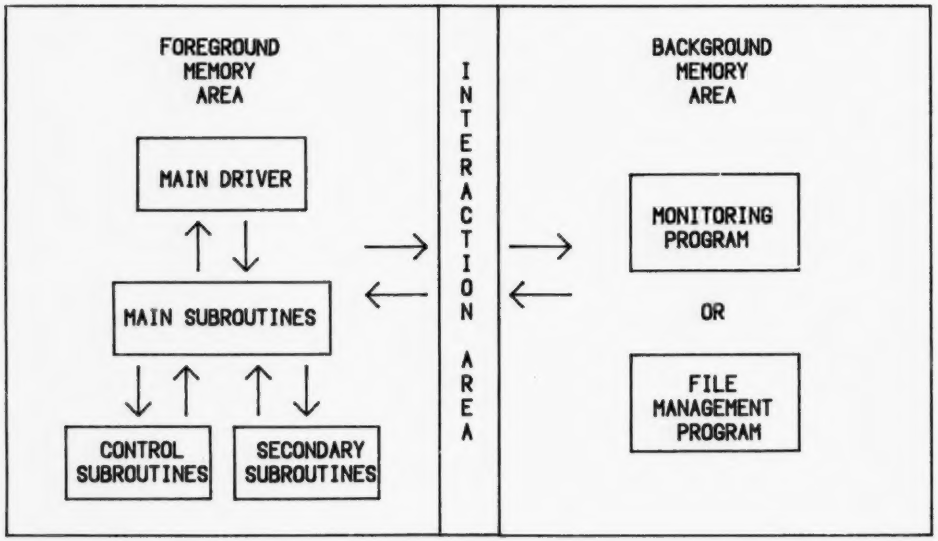


Figure 12. Foreground-background memory and interaction area

TABLE 22. BYTE LOCATIONS FOR PROGRAM INTERACTION

VARIABLE	LOCATION*	FUNCTION
JBDIO	170	input/output device
JBDPGM	171	program execution byte
JBDS1	172	main subroutine
JBDS2	173	control or secondary subroutine
JBDCH1	174	channel to be acted upon
JBDCH2	175	override shutdown
JBDCTL	176	control subroutine location
JBDERR	177	error byte

* Octal number.

TABLE 23. MAIN SUBROUTINE DATA ASSIGNMENTS FOR JBDS1

VARIABLE	VALUE*	SUBROUTINE
JBDAL	10	BOUDAL (phone dialing)
(+1)	11	passed timer
(+2)	12	printed number
(+3)	13	looks for carrier
(+4)	14	test carrier
(+5)	15	succesful attach
JBTMP	20	BOUTMP (temperature)
JBWND	30	BOUWND (wind speed)
JBCON	40	BOUCON (conductivity)
	41	BOUOSC (oscillator test)
JBPRS	50	BOUPRS (commercial conductivity)
JBFFT	60	BOUFFT (flow rate)
	61	BOUDAC (digital-analog converter)
JBERR	70	BOUER1 (error detection)
	71	BOUER2 (shutdown)
JBMAH	80	BOUMH1 (calcs ave cond.)
	81	BOUMH2 (calcs ave readings)
	82	BOUMHT (calcs ave temp)
JBPMP	90	BOUPMP (manipulates pumps)
	91	BOUPDC (checks sync of pumps)
	92	BOUPFT (checks sync of relays)
	93	BOUPFD (turns pumps off if error)
JBSTA	100	BOUSTA (outputs current status)
JBTIM	110	BOUTIM (calcs time pumps on)
JBDK1	120	BOUDK1 (outputs info to disk files)
	121	BOUDK2 (daily ave file operations)
JBSLP	130	BOUSLP (calcs time ref for sleep)
JBPKC	140	BOUPKC (reset diag bytes)
JBINT	150	BOUIN1 (sets operating conditions)
	151	BOUIN2 (sets operating conditions)
JBFAL	250	BOUFAL (exit program restart from bg)

*Decimal number.

TABLE 24. SECONDARY SUBROUTINE DATA ASSIGNMENTS FOR JBDS2

VARIABLE	VALUE*	SUBROUTINE
JBRTL	10	BOURTL (sets cond so data read)
JBCHN	20	BOUCHN (decimal channel to octal)
JBRLL	30	BOURLL (open/close scanner channel)
JBWRT	40	BOUWRT (reads information at site)
JBRET	50	BOURET (channel command sent)
JBRED	60	BOURED (disk operations)
JBBIT	70	BOUBIT (calcs bit status from words)

*Decimal number.

TABLE 25. MISCELLANEOUS DATA ASSIGNMENTS OF VARIABLES

VARIABLE	VALUE*	LOC.	FUNCTION
JBDISK	0	JBDIO	output to disk
JBTT	10	JBDIO	output to terminal
JBECON	1	JBDERR	error oscillator
JBEINT	2	JBDERR	error init disk read
JBETMP	4	JBDERR	error conductance board
JBEFFT	8	JBDERR	error DAC
JBEDAL	16	JBDERR	error phone
JBEHND	32	JBDERR	error handshake
JBEADD	64	JBDERR	error on backup write
JBESHT	128	JBDERR	error shut down
JBOVR	1	JBDCH2	override shut down
JBWAIT	10	JBGPGM	wait mode before starting
JBRUN	20	JBDPGM	program running
JBSLEP	30	JBDPGM	sleep period

* Decimal number.

INTERNAL TRACE BACK FEATURE

The foreground program, along with giving diagnostic support with the background program, has an internal trace back system which, in the case of equipment or timing failure, traces the hierarchy at time of failure. Each routine updates a LOGICAL*1 array which is common with every subroutine in the program. This way the program can be traced back both for failure in timing with the data collection equipment and to check the data as the program is running. BOUCHK will print out the common area if the the source program is compiled with the debugging feature set. All source lines which have a 'D' in column 1 are compiled into the code if the debugging feature is set. If the debugging feature is not set, these lines are just considered to be comments. The call to BOUCHK is prefaced by a 'D' in the main driver, so for normal operations it is ignored.

LAND TREATMENT CONTROL AND MONITORING PROGRAMS

INTRODUCTION

There are actually three programs involved with the control of the land treatment system. They are:

1) BOULD.REL	-	Foreground program that monitors and controls the Land Treatment site.
2) BOURST.SAV	-	Backgound program which monitors the status of Foreground program. This program will restart BOULD.REL if for some reason it terminates or gets hung in a loop. An example of this would be the loss of the carrier or unclean telecommuniction lines.
3) BOUFIL.SAV	-	Background program which lets one view the data files used by BOULD.REL

The background programs were integrated so that when one was exited, the next program was automatically loaded into memory and executed. To return to the FG monitor one had to enter the correct password. This was to prevent inadvertent termination of the monitoring program. If the monitoring program was not functioning, the foreground program would not be restarted.

All three programs were developed using a modular approach. Critical data which were used by each module were put into common blocks, so any module could access it. Each routine was written so its function was independent of the other routines. By combining routines through subroutine calls and hierarching, a total function could be accomplished, that is, the conductivity could be read, or the pumps could be turned on.

CONTROL PROGRAM (BOULD.REL)

BOULD.REL is divided into 3 sections. They are: the 1) main driver program, 2) the main or function subroutines and 3) the control or secondary subroutines. The main driver calls each main, or function subroutine as it is needed. The function subroutines are the ones which control each type of sensor or do a specific calculation involved with the collected data. The control subroutines are the routines which control the H-P data collection system. The secondary subroutines are oriented to input/output and date-time calculations. See Figure 12.

This program was developed so that if the equipment was not working properly, error recovery routines would be executed. Depending upon the severity of the error or equipment failure, the system would perform recovery routines or totally shutdown. If equipment failure was detected in critical modules, the program went into a shutdown mode which then turned off all pumps and relays until the equipment was repaired. Noncritical module failure was logged into an error report file for maintenance purposes and, if needed, predefined default values would be used instead of actual data.

An example of a critical hardware module was the conductivity bridge. If this module failed there was no way to determine the soil moisture, thus no way to control the system. A noncritical module would be the wind speed module. If this failed the treatment system could still operate, but we would only lose the wind blow factor which is dicussed later.

The program was too large to fit into the computer memory, so it was overlayed into three sections. Each overlay section can be called from the previous overlay section, but cannot call any subroutine in the concurrent section.

Sometimes to keep a given overlay section as small as possible, a routine would be broken or written in two or more sections. Depending upon the routine, these sections may be independent of each other. In other cases they were position dependent, such as the error handling routine which had to be executed in order. By using overlays, the only contraints are on the sizes of the common blocks for data storage. The output map from the task building processes lists each subroutine and the amount of memory it requires (Appendix V).

Data files were not accessed by the FORTRAN IV open/close statements but by using RT-11 system library routines. The FORTRAN IV methods used too much system overhead or buffer space for both the sequential and direct access file operations (Digital Equipment Corporation, 1980). The advantage of using RT system calls was the overhead for file I/O could be kept at a minimum as the files were stored binarily, and any record could be accessed in any order. The disadvantage is that the user, not the FORTRAN OTS (operating time system), has to keep track of the next record to be written. The user has to realize that if the file is improperly named, one could inadvertently write over an entire disk. By using standard device and file names this should never occur. The disk I/O is much faster with this method than with the FORTRAN method, but it stores data as words (2 bytes), so all data must be of the same type (real, integers) (Digital Equipment Corporation, 1981e).

A description of the routines which made up the control program are listed by functional section below. The FORTRAN source code is in Appendix I.

MAIN DRIVER

BOULDM is the main driver program. This routine calls all the main subroutines and executes the wait or sleep period before the next reading. The waiting period between cycles for data collection and control could be dynamically changed.

BOUDAT contains all the data in common with each routine. This is a FORTRAN BLOCK DATA file. This file also contains valuable comments and calculation sequences for determining the setpoints and acceptable ranges.

MAIN SUBROUTINES

BOUCHK is used as a dignostic tool when debugging the program. All data in the common block are dumped to the FG terminal if this routine is called.

BOUCON starts the sequence which reads the conductivity sensors in the plots. The conductivity module is also checked to make sure it is operating within specifications. If the module is out of specification, an error flag is set. This error flag is later tested in BOUER1. A failure in the conductivity module is a fatal error and will cause the program to terminate after it enters the shutdown mode.

BOUDAC reads the Digital-to-Analog (DAC) convertor to determine the status of the main and plot pumps. Manual and computer operation can also be determined by analyzing the bit code returned. The reading from the DAC is converted from a decimal to a binary equivalent. This binary number tells which pumps are actually on. Sometimes the scanner relays had chatter which caused the flip/flop chips to be in the opposite state than they indicated. The DAC reading gave the correct state of the pumps and relays.

BOUDAL dials the telephone at the field site in Boulder. This is accomplished by passing a phone number in a variable length ASCII byte string. That phone number is then dialed by the CCU dialer and modem. Control is returned to the main program after a carrier is detected. If the carrier is not detected within a given length of time, the program will exit and restart again. The telephone communications were found to be very susceptible to interferences and power flucuations.

BOUDK1 is the first part of the disk writing routine. This section writes the data which must be updated every time the system was sampled and writes it to the appropriate data files. See BOUDAT (BLOCK DATA) Appendix I for the appropriate files.

BOUDK2 is the second part of the disk writing routine. This section is called only when the current date is changed. The days data are averaged when appropriate and stored in a daily average file. The file pointer is then incremented to the next record of the file.

BOUER1 is the first part of the error detection routine. Various options are performed depending on the error encountered. These are in the form of informative messages and/or system shutdown at the site and program termination until repairs are made. This section determines which, if any, signal conditioning modules are bad. If any are found to be faulty, BOUER2 is executed. If everything is working properly, then BOUER2 is skipped.

BOUER2 is the second part of the error detection routine. This section determines if the system shut down mode should be executed. If so, system shutdown is attempted. First, it is determined if the system shutdown mode is to be overidden. This override is accomplished through the BG program with the FG-BG interaction mode. If the override is not set, then an attempt is made to turn off all pumps and relays. If the failure was not caused by the scanner or CCI then this should work. All pump flags and status indicators are initialized, and the current status is written to the disk. If an equipment problem occurs or a piece of equipment does not meet its specification, the program is normally set to terminate. The override byte prevents this termination. This is especially important in an initial testing stage where the equipment does not always perform as one would expect.

BOUFAL is executed only when the program goes into a timing failure. Timing failures are either telephone related (no answer, CCI bad) or transmission line related (bit dropped). The routines currently being executed are logged on to the job terminal, the phone is disconnected, and the program termination byte is set so the program can be restarted. A code is passed to this routine which tells the reason for failure.

BOUFFT starts the sequence which determines the flow rate to the individual plots. The flow rates are read directly from the flow transducer and needs no conversion.

BOUIN1 is the first of three initializer routines. This section is read only when initially starting or restarting the program. In this section the device handler for the disk files is verified to be loaded. If it is not loaded, it is fetched and loaded. The current readings are read into memory and all status flags are set to the current status. If there was an error on reading the current status file, the backup status file is read. If there is an error reading the backup file, the system logs this on the job terminal and sets a flag so the pump status cannot be changed.

BOUIN2 is the second part of the initializer routines. This section is also only executed when starting or restarting the program. First, the date and time of the control program's next execution is read, and the time to wait for the program to sample the data is calculated. The program either waits this period of time or starts immediate execution of the program if the required time has past. A status byte is sent to the BG program to indicate if the program is waiting or if immediate execution is to take place. Next the system configuration word is read to determine which system the program is running on. This program was also written to run on a PDP 11/03 as a backup computer if the PDP 11/23 failed for a particular reason. Each computer had different addresses for the job terminal and telephone modem so these assignments were then changed. The monitor was also checked to make sure that the program was operating under the proper monitor.

BOUIN3 is the third section of the initializer routines. The setpoints, wait period, time delays, and values to use for minimum acceptable temperature and wind speed, along with the default temperature to use if the temperature board fails are read from a disk file. This data can be changed by using the file management program. If an error occurs reading this file, the default values specified in the BOUDAT are used. This routine is called every time the program runs, not just at startup.

BOUMH1 averages the plot data to determine if the plot pumps and/or relays need to be turned on. If deviations in readings are detected, they are logged for maintenance. Depending on the severity of the deviations, the pumps are either turned off if already on, or not turned on until the problem is corrected. Readings out of the specification range for the conductivities causes the conductivities from the plot to default to the setpoint at which control is being held. This will prevent the pumps and relays from turning on. The conductivity probes are checked to see if the readings are within a specified range of each other. Any probe which fails this test is logged and not used in the calculations to determine the average conductivity.

BOUMH2 stores the current pump status data in the proper locations. The wind speed is checked to determine if its velocity is low enough to apply wastewater. If the wind speed is too high, a flag is set so the pumps cannot be turned on. This prevents inadvertant blow off from the site. The air temperature is checked to determine if it is warm enough to apply wastewater. If the temperature is too low, a flag is set to prevent the wastewater application. The current conductivity readings are then checked against the setpoints. If the current reading is less than the setpoint, the flag to turn on a given plot relay is set. If too many probes are out of range, as determined in BOUMH1, then that plot's relay cannot be turned on.

BOUMHT calculates the average temperature per plot by taking the mean of the temperature transducers in the plot. The transducer readings are compared to each other to determine if they are within specified limits. A weighted average is calculated by taking the readings which are in the specified limits. This average is used in calculating the temperature corrected conductivity readings. If too many probes are out of range, then the average temperature is set to that specified in the file BSTATS.DAT. Any readings which are out of range are logged on the output device. If the average temperature is less than that specified in BSTATS.DAT, a flag is set which prevents the pumps from turning on. This is to prevent freezing of the wastewater distribution system.

BOUOSC reads the oscillator in the conductivity and tests it to verify that the reading is in a specified range. If the reading is out of range, an internal error flag is set, and the background program is also informed of the error.

BOUPDC reads the DAC to determine current pump status. If the wrong pumps are turned on or off, this routine reestablishes the proper synchronization. Manual operation of the system is also noted on the output device. If sychronization cannot be reestablished in a given amount of time, an error flag is set and control is passed to a section which continuously tries to turn the pumps off.

BOUPDF is similar to BOUPDC except pump synchronization is continuously tried until reestablished. The program timer is canceled during this process.

BOUPFT reads the flow transducer to verify that the plot relays were turned on. If the relays cannot be turned on due to some error, an internal flag is set, and the error is logged on the output device.

BOUPKC clears foreground/background interaction bytes which communicate information to the background program. The following bytes are not reset because they are information bytes from the background program to the foreground program: JBDIO and JBCH2. Byte JBDPGM is reset in another section as it informs the background program that the foreground program is still executing. This byte is only set when the foreground program abnormally terminates.

BOUPMP determines the proper sequence to turn the pumps and relays on or off. The pumps and relays are not on a standard on-off type of control but are driven by J/K flip flops which permit clearing the channel scanner (which opens the switch contact) without turning off the pump. The second scanner switch closure opens the relay which operates the individual pump. The current pump status is analyzed to determine which pumps are on and which ones need to be on or off. Only these pumps and relays are acted upon.

BOUMH2 stores the current pump status data in the proper locations. The wind speed is checked to determine if its velocity is low enough to apply wastewater. If the wind speed is too high, a flag is set so the pumps cannot be turned on. This prevents inadvertant blow off from the site. The air temperature is checked to determine if it is warm enough to apply wastewater. If the temperature is too low, a flag is set to prevent the wastewater application. The current conductivity readings are then checked against the setpoints. If the current reading is less than the setpoint, the flag to turn on a given plot relay is set. If too many probes are out of range, as determined in BOUMH1, then that plot's relay cannot be turned on.

BOUMHT calculates the average temperature per plot by taking the mean of the temperature transducers in the plot. The transducer readings are compared to each other to determine if they are within specified limits. A weighted average is calculated by taking the readings which are in the specified limits. This average is used in calculating the temperature corrected conductivity readings. If too many probes are out of range, then the average temperature is set to that specified in the file BSTATS.DAT. Any readings which are out of range are logged on the output device. If the average temperature is less than that specified in BSTATS.DAT, a flag is set which prevents the pumps from turning on. This is to prevent freezing of the wastewater distribution system.

BOUOSC reads the oscillator in the conductivity and tests it to verify that the reading is in a specified range. If the reading is out of range, an internal error flag is set, and the background program is also informed of the error.

BOUPDC reads the DAC to determine current pump status. If the wrong pumps are turned on or off, this routine reestablishes the proper synchronization. Manual operation of the system is also noted on the output device. If sychronization cannot be reestablished in a given amount of time, an error flag is set and control is passed to a section which continuously tries to turn the pumps off.

BOUPDF is similar to BOUPDC except pump synchronization is continuously tried until reestablished. The program timer is canceled during this process.

BOUPFT reads the flow transducer to verify that the plot relays were turned on. If the relays cannot be turned on due to some error, an internal flag is set, and the error is logged on the output device.

BOUPKC clears foreground/background interaction bytes which communicate information to the background program. The following bytes are not reset because they are information bytes from the background program to the foreground program: JBDIO and JBCH2. Byte JBDPGM is reset in another section as it informs the background program that the foreground program is still executing. This byte is only set when the foreground program abnormally terminates.

BOUPMP determines the proper sequence to turn the pumps and relays on or off. The pumps and relays are not on a standard on-off type of control but are driven by J/K flip flops which permit clearing the channel scanner (which opens the switch contact) without turning off the pump. The second scanner switch closure opens the relay which operates the individual pump. The current pump status is analyzed to determine which pumps are on and which ones need to be on or off. Only these pumps and relays are acted upon.

BOUPRS is similar to BOUCON except a commercial conductivity probe is used.

BOUSLP determines the time to wait before the data collection program is to start again. The time to wait is stored in the file BSTAT.DAT. The new time and date for execution are calulated from this value and the current system date and time. The new date and time are stored in the file BPOLLD.DAT in case of a system failure at which time it can be recalled. This wait time is also assigned to variables which are used in the ISLEEP system subroutine.

BOUSTA outputs the current status information to the output device. The temperature, wind speed, flow readings, conductivity readings, pump status, and setpoints are listed. Averages as well as individual readings are listed. Conductivities are in microsiemens, temperatures in degrees centigrade, and wind speed in km/hr.

BOUTIM calculates the amount of time each pump is on. The date and time the pumps were turned on are compared to the present date and time. This information is then stored in a data array for disk storage.

BOUTMP starts the sequence which reads the temperature sensors in the plots. The temperature module is also checked to make sure it is operating within specifications. Signal conditioning interface failures or readings out of the specification range for the module cause the temperatures from the plot to default to the temperature specified in BSTATS.DAT. An error flag is also set which is tested in BOUERR.

BOUWND starts the sequence which reads the wind speed at the Boulder site. The program will temporarily suspend operations when the wind exceeds that specified in BSTATS.DAT.

SECONDARY AND CONTROL SUBROUTINES

BOUBIT converts a decimal number to its binary equivalent. This binary equivalent represents a bit pattern from either a location in memory or the DAC board.

BOUCHN enters into the proper memory address the channel number which is to be opened or closed. This channel number is converted from a single or double digit base ten number to two distinct octal digits. An error flag is returned from this subroutine if the channel operation did not occur. This information is also sent to the background program.

BOUDTC calculates the number of days difference between two dates. It is possible for a system failure to leave the pumps on for more than one day. This is used to determine the length of time that the pumps were on. Leap years are taken into consideration.

BOUDTI converts a REAL*4 date into three distinct integers. These integers are used in date-time calculations. Dates and times are stored as REAL*4 numbers to be compatible with the rest of the data in the disk files. (See section on control program: BOULD.REL)

BOUDYR calculates the next day of the year in a REAL*4 format. Leap years and end of year turnover are taken into consideration.

BOURED is used for file reading and writing on the disk. Dummy variables are passed to this subroutine when the designated file has either read, write, or read/write operations performed on it. An error flag is also returned which indicates that an error occurred and the severity of the error. All disk data file storage uses this routine.

BOURET sends a carriage return to the scanner which tells it that all commands and channels have been sent and to close the channels. If the channels are not closed properly, an error flag is returned.

BOURLL sends conditions so the scanner channels can be closed. An error flag is returned if the commands are not accepted. See the section on Programming the HP-IB bus for the sequence to program the scanner.

BOURTL sends conditions so the data measurement system can be accessed. An error flag is returned if the commands are not accepted. See the section on Programming the HP-IB bus for the sequence to program the data measurement system.

BOUWRT is the subroutine which reads the data from the Hewlett-Packard Measuring system. This data is converted from a ASCII string to a REAL*4 number and then stored in a data array where further data manipulations and conversions can take place.

BACKGROUND MONITOR PROGRAM (BOURST.SAV)

INTRODUCTION

The background monitoring program gives a dynamic update of the foreground program status on a video terminal screen. This update is a routine by routine location of the foreground program. A listing of the FORTRAN codes for the routines is in Appendix II.

MAIN DRIVER

BOURST is the main driver for the background monitoring program. This routine inhibits the normal control C abort feature of the RT-11 system. This program can only be exited through the commands "EXIT" or "QUIT". The JSW (job status word) is also set to allow single character input without waiting for a terminator (Digital Equipment Corporation, 1981f). This is used because the program is constantly checking the bytes in the interaction area for information from the foreground job. If the program had to wait for a command, it is possible that the foreground program could exit without the background program noting this abnormal end. If this happened, the foreground program would not be restarted. The override byte is also set in with routine. Commands are entered one byte at a time until a carriage return is encountered or a maximun number of bytes are encountered. The command string is then checked against the list of valid commands. If the command is not a valid command, the command string is cleared and byte rechecking resumes. If the command is a valid command, the command is executed, the command string is cleared, and byte rechecking resumes. Exiting this program automatically starts the file maintenance program. If the foreground program is executing (not in a sleep mode), the location and status of each subroutine is displayed on the screen. This display is dynamic as it is constantly updated as the program executes. If the foreground program is not executing, the display is cleared, but the program still checks the interaction bytes for an indication as to when the foreground program starts execution again.

MAIN SUBROUTINES

BOUHDR prints the labels for the screen information. These labels are formatted on the screen at predesignated locations.

BOUFLL causes the background program to exit and tries to restart the foreground program. The JSW is reset to normal, then the cause of the failure is displayed before the foreground restart command is sent to the system monitor.

BOURED is the file manipulation routine. This is discussed under the foreground program.

BOUHLP is the help file which is displayed on the screen. In the normal mode this file is only displayed for a few seconds before byte rechecking resumes. It is possible to display this file longer but to do so inhibits byte checking.

FILE MAINTENANCE PROGRAM (BOUFIL.SAV)

INTRODUCTION

The file maintenance program is the background access to the foreground data files. This program also can change some of the operating parameters and setpoints of the foreground program. A listing of the FORTRAN codes for the routines is in Appendix III.

MAIN DRIVER

BOUFIM is the main driver for the program BOUFIL.SAV. This routine prompts for the file to be viewed or manipulated; then one must verify that this is the correct file, and the file is then displayed on the terminal screen. File selection is determined either by stepping through the file list one file at a time or by going directly to the file by entering its number. Exiting this program causes an automatic restart of the monitoring program.

MAIN SUBROUTINES

BOUFIS allows the user to change the data in one of the data files. This module is entirely screen driven with one record of the file displayed at a time. A write protection code is needed to be able to change data. This routine is mainly used to change the input data to BOULD.REL from the file BSTATS.DAT.

BOUFIR outputs a selected range of nonblank records of the specified file to the output device. A nonblank record is a record which is not all zeros. Initially, all data files are built or created to have blank records.

BOUFID outputs all nonblank records of the specified file to the output device.

BOUFIH is the help file for this program. This help file is displayed on the screen. The output device for data dumps is also listed.

BOUFIW initializes the specified file to all zeroes or all blank records. A write protection code must be entered before this process will take place.

BOUFIP reads and outputs only a maximum of 10 fields of data from each record to the output device. Blank records are not ignored. The locations in the data array of the fields to be listed are entered one at a time.

BOUFIN creates a file of specified length and number of words. The created file is a binary file and not a file created by FORTRAN IV (Digital Equipment Corporation, 1981e).

BOUFIF lists the functions of each file on the screen. The names are printed by the main driver.

CONVRT converts a LOGICAL*1 or BYTE string, to a REAL*4 number. This number is accurate for 6 places on either side of the decimal point. If the string cannot be converted to a number, an error flag is returned which specifies that this is a string and not a number. This routine is used in the main driver to determine which file is to be accessed.

BOURED is the file manipulation routine which is fully discussed under BOULD.REL.

TIMLIB holds the initial time and date calculations function. The system date and time were accessed using this group of subroutines. The specific routines are discussed below.

TIMDIF is a function routine which calculates the difference between two times. The new time is the first variable and the old time is second variable. Both times are in REAL*4 format and the result is a REAL*4 number.

SYSTIM accesses the system data and time and converts them to REAL*4 numbers.

SYSTMC returns the minutes in the time to its decimal equivalent. The time 9:30 becomes 9.50.

TIMADD adds two REAL*4 times together and returns a flag which tells if this time is in the next day. There are 60 minutes in a hour so times cannot be directly added.

VIDEO ROUTINES

All the video handler routines are prefaced by 'VT' and then a mnemonic of their function. These mnemonics resemble as closely as possible those specified in the VT100 users manual (Digital Equipment Corporation 1979c).

VTDHLB sets the VT100 to accept the bottom half of a double height line, This is the same text as sent with VTDHLT and must be synchronized with it.

VTDHLT sets the VT100 to accept the top half of a double height line. This is the same text as sent with VTDHLB and must be synchronized with it.

VTED erases the specified area of the display. If an 'A' is specified, the whole display is erased. If a 'E' is specified, the display is erased from the current cursor location to the end of the display. If a 'B' is specified, the display is erased from the current location to the beginning of the display. In all cases the cursor is left in its current position.

VTERSE erases the whole screen and returns the cursor to home position. This is used to clear the screen and start printing at the top of the screen.

VTHVP positions the cursor at a specified horizontal and vertical position. The row and column are sent to the screen and the cursor is positioned at that point.

VTSGR selects the type of graphic rendition to display on the screen. The graphic renditions are reverse video (R), underline (U), blinking (B), and bold or halfbright (H). The type of rendition is accomplished by sending a R, U, B, or H.

VTTBM sets the top and bottom margins of the scrolling region of the screen. The default scrolling region is from line 1 to line 24. The minimum scrolling region can be as small as any two consecutive lines on the display.

VTWID changes the width of the screen to 80 or 132 columns, and vice versa.

VTDWL causes the next text on that line to double width instead of single width.

VTEL erases the specified area of the current line. If an 'A' is specified, the whole line is erased. If a 'E' is specified, the current line is erased from the cursor location to the end of the line. If a 'B' is specified, the line is erased from the cursor to the beginning of the line. In all cases, the cursor is left in its current position.

COMMAND FILES TO BUILD AND OPERATE SYSTEM

The following is a brief function of the command files to build the Boulder data collection, file maintenance and monitoring programs. The '@' character in front of the name implies that this type file is a command file with a file type of '.COM' and can be executed directly by typing the file name as specified below. The command file made it easy to maintain and update the computer code for the total data collection system. The whole software system could be rebuilt in a matter of minutes. A list of the command files is in Appendix VI.

@TIMLIB builds the time calculation library TIMLIB.OBJ.

@BOULDA builds the foreground program BOULD.REL. This file calls two other command files: @BOULDF (FORTRAN source complilation) and @BOULDL (task builder and overlay).

@BOUFIA builds the program that reads and manipulates the data files. The resultant program is BOUFIL.SAV.

@BOURSA builds the status and restart program BOURST.SAV. This file calls two other files: @BOURSF (FORTRAN source compilation) and @BOURSL (task builder and overlay).

@VTLIB builds the video handler library VTLIB.OBJ.

@BOULD starts the foreground program and then returns to the monitor.

@RESTRT starts both @BOULD and BOURST.SAV.

@BFILE starts BOUFIL.SAV when exiting BOURST.SAV.

@BSTRT starts BOU:BOURST.SAV when exiting BOUFIL.SAV.

REFERENCES

ALTOMETER. ALTOFLUX MT 900 Magnetic-Inductive flowmeter. B.V. Altometer Sliedrecht/Netherlands Postbus 46.

Digital Equipment Corporation. 1977. DLV11-E and DLV11-F Asynchronous Line Interface Users' Manual. EK-DLV11-OP-001. Maynard, MA.

Digital Equipment Corporation. 1979a. Microcomputer Processor Handbook. Maynard, MA.

Digital Equipment Corporation. 1979b. PDP-11 FORTRAN Language Reference Manual. AA-1855-TC. Maynard, MA.

Digital Equipment Corporation. 1979c. VT100 USER GUIDE. EK-VT100-UG-002. Maynard, MA.

Digital Equipment Corporation. 1980. RT-11,RSTE/E FORTRAN IV Users Guide. AA-5749B-TC. Maynard, MA.

Digital Equipment Corporation. 1981a. Microcomputer Interfaces Handbook. Maynard, MA.

Digital Equipment Corporation. 1981b. Microcomputer and Memories. Maynard, MA.

Digital Equipment Corporation. 1981c. RT-11 Installation and System Generation Guide. AA-H376A-TC. Maynard, MA.

Digital Equipment Corporation. 1981d. Introduction to RT-11. AA-5281B-TC. Maynard, MA.

Digital Equipment Corporation. 1981e. RT-11 Programmer's Reference Manual. AA-H378A-TC. Maynard, MA.

Digital Equipment Corporation. 1981f. RT-11 Software Support Manual. AA-H379A-TC. Maynard, MA.

Digital Equipment Corporation. 1981g. RT-11 System Users Guide. AA-5279B-TC. Maynard, MA.

Enfield, C. G., 1977. Servo Controlled Optimization of Nitrification-Denitrification of Wastewater in Soil. J. Environ. Qual. 6:456-458.

Hewlett Packard. 1974. Programming and Service Manual Model 34951 Scanner. Manual part no. 03495-900012. Hewlett Packard Co. Loveland, CO.

Hewlett Packard. 1975a. Operating and Service Manual ASCII Interface 5312A. Manual part no. 05312-90004. Hewlett Packard Co. Loveland, CO.

Hewlett Packard. 1975b. Operating and Service Manual Universal Counter 5302A. Manual part no. 05302-90009. Hewlett Packard Co. Loveland, CO.

Hewlett Packard. 1976. Operating and Service Manual. Revision A Model 59403A Common Carrier Interface. Manual part no. 59403-90001. Hewlett Packard Co. Loveland, CO.

Hewlett Packard. 1977. Operating and Service Manual. Measuring System 5300B and Battery Pack 5310A. Manual part no. 05300-90035. Hewlett Packard Co. Loveland, CO.

Leach, L. E. and C. G. Enfield. 1983. Nitrogen Control in Domestic Wastewater Rapid Infiltration Systems. J. Water Pollut. Control Fed. 55:1150-1157.

Timplex. 1977a. Model CCU-II Computer Calling Unit. Customer Manual. Timplex. Hackensack, NJ.

Timplex. 1977b. Timplex Modem Systems 103 C/D Customer Manual. Timplex. Hackensack, NJ.

APPENDIX I

CONTROL PROGRAMS SOURCE CODE

```
      PROGRAM BOULD
c +-------------------------------------------------------------------+
c | Program :    BOULD.REL                                            |
c | Module  :    BOULD.FOR                                            |
c | Function:    Main driver for boulder land treatment program       |
c |                                                                   |
c +-------------------------------------------------------------------+
c |                                                                   |
c |       See BOUDAT.FOR for documentation and variable assignments   |
c |                                                                   |
c |  See BOULD.BLD for the use of command files to build the needed progr|
c |                                                                   |
c |  NOTE: THE CONVENTION FOR CONTINUATION LINES IS A TAB AND A CHARACTER|
c |         SUCH AS A '*' OR A " ".                                   |
c |                                                                   |
c +-------------------------------------------------------------------+
c--
c--   data common block
c--
      COMMON/BOULD/BOUDAT(60),BOUA(25),BOUB(25),BDATA(60),BOUT(60),
      *NARRAY,BCONAV(2),BTMPAV(2),BPRSAV(2),NPL,NPR,NBT1,NBT2,NBC1,
      *NBC2,NBP1,NBP2,LBCF(2),LBTF(2),LBPF(2),SETPTC(2),LOCB,
      *KFBR(2),KFBP,KSBR(2),KSBP,KTMPF,KCONF,KPRSF,KINTF,KWNDF,KFFTF,
      *KPUMP(2)
c--
c--   channel switching common block
c--
      COMMON/CHAN/IBCONB,IBCOO,IBFCI,IBVFI,IBVFO,IBTCM,IBREFR,
      *IBWS,IBMPR,IBPPR,IBPNIC,MBCON(7),NBCON,MBTMP(10),
      *NBTMP,MBPMP(2),NBPMP,MBPRS(7),NBPRS,MBFFT(2),NBFFT,
      *MBCONR(3),NBCONR
c--
c--   data checking common block
c--
      COMMON/CONDCL/WINDMX,TEMPMN,BCONL,BCONH,BCONRF(3),BRI(3),
      *BVOLTS,BAI,BTMPRF,BCONIN,BAVETP(2),BTMPHZ,RANL,RANH,BFTDAC(4)
c--
c--   control character common block
c--
      COMMON /CONTRL/LA,LB,LC,LR,LDLE,LQU,LLAS,LTACT,LLACT,LTAC,
      *LLAC,LI,LF,LCR,LLF,ITRH1,ITRM1,ITRS1,ITRT1,ITRT2,ITRT3
c--
c--   diagnostic block
c--
      LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
      INTEGER*4 I1AREA,I2AREA
      COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
      *LTERM,ITERM,ATIMES,BTIMES,IADPOK,IADPEK,IPHONE,ILDTR,IHDTR,
      *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
      *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
      *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
      *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
      *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
      *JBOVR,JBWAIT,JBSLEP
c--
c--   file information block
c--
      COMMON/FILE/BOUDEV,MABLK(4),MBBLK(4),MCBLK(4),MDBLK(4),
      *MEBLK(4),MFBLK(4),MGBLK(4),MHBLK(4),MIBLK(4),MJBLK(4),MKBLK(4),
      *MLBLK(4),MMBLK(4),MNBLK(4),MOBLK(4),MPBLK(4),MQBLK(4),MRBLK(4),
      *MSBLK(4),
      *MABLKL,MABLKW,MBBLKL,MBBLKW,MCBLKL,
      *MCBLKW,MDBLKL,MDBLKW,MEBLKL,MEBLKW,MFBLKL,MFBLKW,MGBLKL,
      *MGBLKW,MHBLKL,MHBLKW,MIBLKL,MIBLKW,MJBLKL,MJBLKW,MKBLKL,
      *MKBLKW,MLBLKL,MLBLKW,MMBLKL,MMBLKW,MNBLKL,MNBLKW,MOBLKL,
      *MOBLKW,MPBLKL,MPBLKW,MQBLKL,MQBLKW,MRBLKL,MRBLKW,MSBLKL,MSBLKW
c--
c--   telephone number common block
c--
      LOGICAL*1 NUMBOU,NUMADA
      COMMON/NUMBER/NUMBOU(13),NUMADA(9)
c--
c--   base address common block
c--
      COMMON/PROCES/IAD103,IAD123,ITM03,ITM23
c--
c--   timer failure common block
c--
      COMMON/TIMER/ITMH,ITMM,ITMS,ITMT,ITDH,ITDM,ITDS,ITST
c--
c--   main program sleep block
c--
      COMMON/WAIT/IBPOLH,IBPOLM,IBPOLS,IBPOLT,IBCC,IBCH,IBCM,
      *IBCS,IBCT
c--
c--
c--   bits common
c--
      COMMON/BITS/IBIT(16),IWORD
c--
c--   <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <>
c--
      EXTERNAL FAIL
c--
c--   wait for bg program monitoring
c--
      CALL ISLEEP (0,0,10,0)  !SYNC WITH BG TESTING
      CALL ASSIGN (LTERM,'TT:')
      CALL BOUPKC      !RESET DIAGNOSTIC BYTES WITH BG
c--
c--   get time and date before anything else
c--
      SYSTM1=SYSTIM(SYSDAT)   !get system date and time (real*4)
c--
```

```
c--     check for proper date
c--
        IF (SYSDAT.GT.830000.) GOTO 500
        DO 535,I=1,10
535     WRITE (LTERM,534) "007
534     FORMAT(X,1A1)
        CALL IPOKEB (JBDPGM,1)  !set date failure no run
        WRITE (LTERM,*) 'SYSDAT',SYSDAT
        STOP 'INVALID DATE CAN NOT RUN'
c--
c--     timing failure traceback
c--
500     CALL SCOPY (' BOULD',LEVEL1)
        CALL SCOPY ('      ',LEVEL2)
        CALL SCOPY ('       ',LEVEL3)
c--
c--     tempory diagnostics for debugging
c--
D       CALL PUTSTR(LTERM,' LOCATION: ','0')
D       CALL PUTSTR(LTERM,LEVEL1,' ')
D       CALL PUTSTR(LTERM,LEVEL2,' ')
D       CALL PUTSTR(LTERM,LEVEL3,' ')
c--
c--     make sure ther are enough queue elements
c--
        IF ((IQSET(5)).NE.0) WRITE (LTERM,333)
c--
c--     set initial values and determine which processor
c--
        CALL BOUIN1      !SETS INITAL DATA VALUES
        CALL BOUIN2
c--
c--     ibcc is flag to tell if this is first time thru
c--
        IF (IBCC.NE.0) WRITE (LTERM,98) IBCH,IBCM
        CALL IPOKEB (JBDPGM,JBWAIT)     !wait mode
        IF (IBCC.NE.0) GOTO 99  !TOO SOON TO RUN PROGRAM
c--
c--     find out where output is supposed to go
c--
c--     this is set in the restrt program running in background
c--
        CALL CLOSE(LTERM)
10      CALL IPOKEB (JBDPGM,JBRUN)      !run mode
        IF (IPEEKB(JBDIO).EQ.JBDISK)
        *       CALL ASSIGN (LTERM,'BOU:BOULD.DAT/C')
        IF (IPEEKB(JBDIO).EQ.JBTT)
        *       CALL ASSIGN (LTERM,'TT:')
c--
c--
c--     read set points and other data
c--
```

```
        CALL BOUIN3
c--
c--     following 2 lines set and test timer to make sure their
c--     is a normal return. if communications takes longer than that defid
c--     by the timer subroutine fail is called.
c--
        IF( ITIMER (ITMH,ITMM,ITMS,ITMT,I1AREA,1,FAIL).NE.0)
          WRITE(LTERM,334)
        BTIMES=BTIMES+1 !number of times thru
        WRITE(LTERM,20) BTIMES
        SYSTM1=SYSTIM(SYSDAT)   !get system date and time (real*4)
        SYSDAT=SYSDAT
        WRITE (LTERM,21) SYSTM1,SYSDAT
c--
c--     following lines call subroutines which gather and store the
c--     boulder data
c--
        CALL DIALT(NUMBOU)      !PHONE DIALING SUBROUTINE
c--
        CALL BOUTMP     !READ TEMPERATURES
c--
D       CALL BOUCHK     !******** DUMP DATA ARRAYS FOR VERIFICATION
c--
        CALL BOUMHT     !CALCULATE AVE. TEMP
c--
        CALL BOUWND     !GET WIND SPEED
c--
D       CALL BOUCHK     !***************
c--
        CALL BOUOSC     !check oscillator before reading cond.  comm cond
c--
D       CALL BOUCHK     !***************
c--
        CALL BOUCON     !READ CONDUCTIVITIES
c--
D       CALL BOUCHK     !***************
c--
        CALL BOUPRS     !READ COMMERCIAL CONDUCTIVITIES
c--
D       CALL BOUCHK     !***************
c--
        CALL BOUDAC     !READS DAC CONVERTOR
c--
D       CALL BOUCHK     !***************
c--
        CALL BOUFFT     !READ FLOW RATE
c--
D       CALL BOUCHK     !***************
c--
        CALL BOUER1     !CHECK MODULES---SHUT DOWN?
c--
D       CALL BOUCHK     !***************
```

```
c--
        CALL BOUER2
c--
D       CALL BOUCHK     !***************
c--
        CALL BOUMH1
c--
D       CALL BOUCHK     !***************
c--
        CALL BOUMH2
c--
D       CALL BOUCHK     !****************
c--
        CALL BOUPMP
c--
        CALL BOUPDC(IEQUIP)     !CHECKS DAC FOR PUMP SYNC
c--
        CALL BOUPFT(IEQFFT)             !CHECKS FLOW FOR PLOT RELAY OPERAN
c--
        CALL IPOKE (IPHONE,ILDTR)       !CAUSES PHONE TO DISCONNECT
c--
        CALL ICMKT(1,I1AREA)    !CANCELS TIMER
c--
        CALL BOUTIM     !calculate how long pumps where on
c--
        CALL BOUSTA     !OUTPUT STATUS INFORMATION
c--
        CALL BOUDK1     !wirte part of information to disk
c--
        CALL BOUDK2     !write rest of information to the disk
c--
        CALL BOUSLP     !calculate amount of time to sleep
c--
        CALL BOUPKC     !RESET DIAGNOSTIC BYTES
c--
c--     bg sleep indicator
c--
        CALL IPOKEB (JBDPGM,JBSLEP)     !sleep period
c--
c--     close ouput file last thing beforre sleep
c--
99      CALL CLOSE (LTERM)
        CALL ISLEEP (IBCH,IBCM,IBCS,IBCT)
        GO TO 10        !go to 10 and repeat program
c-------------------------------------------------------------------------
c--
c--     format statements
c-------------------------------------------------------------------------
20      FORMAT (/,'1TIMER RESET  TIMES THRU= ',F10.0)
21      FORMAT (' TIME :',F5.0,10X,'DATE :',F7.0)
98      FORMAT (' PROGRAM IS NOT TO RUN FOR ANOTHER',
        *I3,' HOURS',I4,' MINUTES')
```

```
333     FORMAT (' NOT ENOUGH FREE SPACE FOR QUEUE ELEMENTS')
334     FORMAT (' TIMER DID NOT HAVE NORMAL RETURN')
        END
        BLOCK DATA
c +---------------------------------------------------------------------+
c | Program :   BOULD.REL                                               |
c | Module  :   BOUDAT.FOR                                              |
c | Function:   DATA BLOCK AND DCOUMENTATION OF VARIABLES               |
c +---------------------------------------------------------------------+
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c|                    Project History
c|
c| Synopsis of project history and problems
c|
c|   DATE                            COMMENT
c| ~~~~~~~~    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
c| 01/03/83    Ready to take control. Boulder wants to manually operate
c|              for a while longer . Continue to monitor and check program
c|              logic.
c| 01/31/83    Problem discovered with accumulator file. File was not
c|              reinitializing to zero
c| 02/19/83    Problem with air compressor at Boulder which operates
c|              remote valves. Controls will not function with out compred
c|              air.
c| 02/28/83    Permanent collection of data started for base line.
c|              (File management system and monitor programs verified.)
c| 03/14/83    Plot wastewater delivery line reconstruction started.
c|              Projected down time 2 weeks.
c|              Program modified to check for failure of contacts closing
c|              properly. Pumps may not be in proper sync.
c|              Board  and software needs modified to correct this.
c|              Program could still run though.
c| 04/05/83    RK05 compatible disk problem with belts and pulleys. New ps
c|              ordered. Temporarily ceasing monitoring until repaired.
c|              Construction on delivery line not complete.
c| 04/25/83    Parts arrived. Fixed disk drive and retarted monitoring.
c|              Construction on delivery line not complete.
c|
C| 05/20/83     construction completed on delivery line completed.
c|              started to check out system. Compressor head cracks.
c|
C| 07/11/83     Ready to check out equipment.Compressor valve ruptures dug
c|               test. Have to install new valve and retest compressor.
c|
c| 08/11/83     Starting to check pump routines to make sure they work pr.
c|
c| 08/15/83     plot drying out for mowing to take place tomorrow
c|
c| 01/06/84     changed probe from plot 2 sta 3 (24) to plot 2 sta 1 (22)
c|
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c*
c*   Note:  There are actulally three programs involved with the control
c*          of the land treatment system. They are:
c*
```

```
c*        1) BOULD.REL     - Foreground program that monitors and controlse
c*                           Land Treatment site.
c*        2) BOURST.SAV    - Backgound program which monitors the status of
c*                           Foreground program. This program will restart
c*                           BOULD.REL if for some reason if terminates.
c*                           ( timing errors from phone lines)
c*        3) BOUFIL.SAV    - Background program which lets one view the daa
c*                           files used by BOULD.REL
c*
c*    If one leaves either Backgroud program by the "EXIT" command the
c*     Background program will start. There are help files in both progra.
c***********************************************************************
c--
c--
c--      data common block
c--
         COMMON/BOULD/BOUDAT(60),BOUA(25),BOUB(25),BDATA(60),BOUT(60),
         *NARRAY,BCONAV(2),BTMPAV(2),BPRSAV(2),NPL,NPR,NBT1,NBT2,NBC1,
         *NBC2,NBP1,NBP2,LBCF(2),LBTF(2),LBPF(2),SETPTC(2),LOCB,
         *KFBR(2),KFBP,KSBR(2),KSBP,KTMPF,KCONF,KPRSF,KINTF,KWNDF,KFFTF,
         *KPUMP(2)
c--
c--      channel switching common block
c--
         COMMON/CHAN/IBCONB,IBCOO,IBFCI,IBVFI,IBVFO,IBTCM,IBREFR,
         *IBWS,IBMPR,IBPPR,IBPNIC,MBCON(7),NBCON,MBTMP(10),
         *NBTMP,MBPMP(2),NBPMP,MBPRS(7),NBPRS,MBFFT(2),NBFFT,
         *MBCONR(3),NBCONR
c--
c--      data checking common block
c--
         COMMON/CONDCL/WINDMX,TEMPMN,BCONL,BCONH,BCONRF(3),BRI(3),
         *BVOLTS,BAI,BTMPRF,BCONIN,BAVETP(2),BTMPHZ,RANL,RANH,BFTDAC(4)
c--
c--      control character common block
c--
         COMMON /CONTRL/LA,LB,LC,LR,LDLE,LQU,LLAS,LTACT,LLACT,LTAC,
         *LLAC,LI,LF,LCR,LLF,ITRH1,ITRM1,ITRS1,ITRT1,ITRT2,ITRT3
c--
c--      diagnostic block
c--
         LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
         INTEGER*4 I1AREA,I2AREA
         COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
         *LTERM,ITERM,ATIMES,BTIMES,IADPOK,IADPEK,IPHONE,ILDTR,IHDTR,
         *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
         *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
         *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
         *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
         *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
         *JBOVR,JBWAIT,JBSLEP
c--
```

```
c--     file information block
c--
        COMMON/FILE/BOUDEV,MABLK(4),MBBLK(4),MCBLK(4),MDBLK(4),
        *MEBLK(4),MFBLK(4),MGBLK(4),MHBLK(4),MIBLK(4),MJBLK(4),MKBLK(4),
        *MLBLK(4),MMBLK(4),MNBLK(4),MOBLK(4),MPBLK(4),MQBLK(4),MRBLK(4),
        *MSBLK(4),
        *MABLKL,MABLKW,MBBLKL,MBBLKW,MCBLKL,
        *MCBLKW,MDBLKL,MDBLKW,MEBLKL,MEBLKW,MFBLKL,MFBLKW,MGBLKL,
        *MGBLKW,MHBLKL,MHBLKW,MIBLKL,MIBLKW,MJBLKL,MJBLKW,MKBLKL,
        *MKBLKW,MLBLKL,MLBLKW,MMBLKL,MMBLKW,MNBLKL,MNBLKW,MOBLKL,
        *MOBLKW,MPBLKL,MPBLKW,MQBLKL,MQBLKW,MRBLKL,MRBLKW,MSBLKL,MSBLKW
c--
c--     telephone number common block
c--
        LOGICAL*1 NUMBOU,NUMADA
        COMMON/NUMBER/NUMBOU(13),NUMADA(9)
c--
c--     base address common block
c--
        COMMON/PROCES/IAD103,IAD123,ITM03,ITM23
c--
c--     timer failure common block
c--
        COMMON/TIMER/ITMH,ITMM,ITMS,ITMT,ITDH,ITDM,ITDS,ITST
c--
c--     main program sleep block
c--
        COMMON/WAIT/IBPOLH,IBPOLM,IBPOLS,IBPOLT,IBCC,IBCH,IBCM,
        *IBCS,IBCT
c--
c--     <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <>
c--
c
c
c                DATA STATEMENTS AND DOCUMENTATION
c
c
c---------------------------------------------------------------------
c--
c--     data for output and main timer functions
c--     main timer in driver determines how long program is to run
c--     before program is to exit and restart. this is usually caused by a
c--     hand shake error.
c--
c--     ..M.    refers to main timer in main program
c--     ..D.    refers to timer in dialing section
c--     ...H    hours
c--     ...M    minutes
c--     ...S    seconds
c--     ...T    ticks
c--
c--     output takes place with LTERM. this is decided in the main program
```

```
c--     by reading one of the diagnostic bytes in locations "170-"177
c--     [options are a disk file BOULDER.DAT or the job terminal]
c--
c-----------------------------------------------------------------------
c--
        DATA LTERM/3/ITMH,ITMM,ITMS,ITMT/0,10,0,0,/
        *ITDH,ITDM,ITDS,ITST/0,5,0,0/
c-----------------------------------------------------------------------
c--
c--     telephone numbers and other address data
c--     numada  number for ada
c--     numbou  number for boulder
c--
        DATA NUMBOU/'1','1','8','0','0','4','5','7','8','0','9','3',"000/
        DATA NUMADA/'1','3','3','2','8','8','1','0',"000/
c--
c--     first "1" is to take phone off hook (TIMPLEX dialer specific)
c--
c--     data for DTR (data terminal ready) connects and hangs up phone
c--     ILDTR   hangs up phone
c--     IHDTR   connects phone so number can be dialed
c--
        DATA ILDTR/"144/IHDTR/"146/
c--
c--     address data for 1103 and 1123
c--     AD1103  base address 1103
c--     AD1123  base address 1123
c--     base addresses are used to calculate IADPOK IADPEK IPHONE
c--     IPHONE=base
c--     IADPEK=base+2
c--     IADPOK=base+6
c--
c--     IAD1103="176500         remote terminal ))) ITM03 = 2
c--     IAD1123="176530         remote terminal ))) ITM23 = 3
c--
c--     these values are used in bouint to determine which proceesor
c--     pdp 1103 or pdp 1123 that the program is running under.
c--     the proper assignments and conditions are then set.
c--
        DATA IAD103/"176500/IAD123/"176530/ITM03/2/ITM23/3/
c-----------------------------------------------------------------------
c--     data to define file names
c--     file names are radix50 format that is
c--
c--     DATA NAME/3RDEV,3RFILE,3RNAM,3RDAT/
c--
c--     which means file name is called DEV:FILNAM.DAT  which follows
c--     name is an integer*2 array   name(4)
c--     file manipulations are done according to those procedures
c--     outlined in  RT-11 PROGRAMMERS REFERENCE MANUAL MARCH 1981
c--     order no. AA-H378A-TC  Chapter 3 System Subrountine Decriptions ad
c--     Examples.
```

```
c--     The following subroutines are used during file operations:
c--
c--             CLOSEC  SECTION 3.3
c--             IENTER  SECTION 3.24
c--             LOOKUP  SECTION 3.74
c--             IFREEC  SECTION 3.26
c--             IGETC   SECTION 3.27
c--             IREOPN  SECTION 3.44
c--             ISAVES  SECTION 3.45
c--             IREADW  SECTION 3.42
c--             IWRITW  SECTION 3.60
c--
c--     The beginning block number of a file starts at block 0.
c--     Do not reference a device only at block 0 as it is possible to thn
c--     distroy the directory segment.
c--
c--     For a real*4 array the number of words is equal to
c--     the array length * 2 that is an array of 60 variables * 2 bytes
c--     per word.   [   60*2=120words   ]
c--
c--     Format for naming variables to be used with file operations
c--     where . = A,B,C,D,etc.
c--     M.BLK(4)        array to store file name
c--     M.BLKL          variable for length of the file in blocks
c--     M.BLKW          variable for length in words of the read/write ary
c--
c--                      FILE NAMES     LENGTH  WORDS   COMMENTS
c--                      ~~~~~~~~~~     ~~~~~~  ~~~~~   ~~~~~~~~~~~~~~~~~
c--     MABLK(4)        BOU:BREADD.DAT  1       120     last reading
c--     MBBLK(4)        BOU:BAVE01.DAT  1       120     acc for hourly ave
c--     MCBLK(4)        BOU:BAVE24.DAT  24      120     staus bytes for BG
c--     MDBLK(4)        BOU:BAVEDL.DAT  90      120     daily ave
c--     MEBLK(4)        BOU:BAVEDR.DAT  1       2       rec num for dailye
c--     MFBLK(4)        BOU:BAVEWK.DAT  7       120     previous 7 days
c--     MGBLK(4)        BOU:BAVEWR.DAT  1       2       rec num for prev y
c--     MHBLK(4)        BOU:BAVEMN.DAT  30      120     previous 30 days
c--     MIBLK(4)        BOU:BAVEMR.DAT  1       2       rec num for prev y
c--     MJBLK(4)        BOU:BPOLLD.DAT  1       4       time to run
c--     MKBLK(4)        BOU:BALLRC.DAT  1       2       rec num for 200 rd
c--     MLBLK(4)        BOU:BALLRD.DAT  200     120     last 200 readings
c--     MMBLK(4)        BOU:BACKUP.DAT  1       120     backup for mablk
c--     MNBLK(4)        BOU:BSTATS.DAT  1       120     set points and vas
c--     MOBLK(4)        ******      EXTRA FILES BELOW HERE      ******
c--     MPBLK(4)
c--     MQBLK(4)
c--     MRBLK(4)
c--     MSBLK(4)
c--
c--     BOUDEV          BOU:            OPERATIONS ON LOGICAL NAME BOU:
c--
c-----------------------------------------------------------------------
        DATA MABLK/3RBOU,3RBRE,3RADD,3RDAT/MABLKL,MABLKW/1,120/
```

```
        DATA MBBLK/3RBOU,3RBAV,3RE01,3RDAT/MBBLKL,MBBLKW/1,120/
        DATA MCBLK/3RBOU,3RBAV,3RE24,3RDAT/MCBLKL,MCBLKW/24,120/
        DATA MDBLK/3RBOU,3RBAV,3REDL,3RDAT/MDBLKL,MDBLKW/90,120/
        DATA MEBLK/3RBOU,3RBAV,3REDR,3RDAT/MEBLKL,MEBLKW/1,2/
        DATA MFBLK/3RBOU,3RBAV,3REWK,3RDAT/MFBLKL,MFBLKW/7,120/
        DATA MGBLK/3RBOU,3RBAV,3REWR,3RDAT/MGBLKL,MGBLKW/1,2/
        DATA MHBLK/3RBOU,3RBAV,3REMN,3RDAT/MHBLKL,MHBLKW/30,120/
        DATA MIBLK/3RBOU,3RBAV,3REMR,3RDAT/MIBLKL,MIBLKW/1,2/
        DATA MJBLK/3RBOU,3RBPO,3RLLD,3RDAT/MJBLKL,MJBLKW/1,4/
        DATA MKBLK/3RBOU,3RBAL,3RLRC,3RDAT/MKBLKL,MKBLKW/1,2/
        DATA MLBLK/3RBOU,3RBAL,3RLRD,3RDAT/MLBLKL,MLBLKW/200,120/
        DATA MMBLK/3RBOU,3RBAC,3RKUP,3RDAT/MMBLKL,MMBLKW/1,120/
        DATA MNBLK/3RBOU,3RBST,3RATS,3RDAT/MNBLKL,MNBLKW/1,120/
        DATA MOBLK/3RBOU,3R   ,3R   ,3RDAT/MOBLKL,MOBLKW/1,2/
        DATA MPBLK/3RBOU,3R   ,3R   ,3RDAT/MPBLKL,MPBLKW/1,2/
        DATA MQBLK/3RBOU,3R   ,3R   ,3RDAT/MQBLKL,MQBLKW/1,2/
        DATA MRBLK/3RBOU,3R   ,3R   ,3RDAT/MRBLKL,MRBLKW/1,2/
        DATA MSBLK/3RBOU,3R   ,3R   ,3RDAT/MSBLKL,MSBLKW/1,2/
        DATA BOUDEV/3RBOU/
c--
c-----------------------------------------------------------------------
c--     boua is used for converting readings to valid data
c--     data values for BOUA array
c-------------------------------------------------------------------------
c--     Formula to calculate temperature is:
c--       degrees centigrade = ((freq/10.)+K1)/K2
c--             K1 and K2 are constants
c--
c--
c--     calibration 20-oct-83
c--
c--     plot    sta     depth   freq.           temp.   constant        1.
c--     ~~~~    ~~~     ~~~~~   ~~~~~           ~~~~~   ~~~~~~~~        ~~
c--     reference               54.8            6       -4.88
c--                                                             -5.84   1
c--     1       1       3       85.37           38.3    -4.71
c--     1       1       3       48.74           0.5     -4.82
c--                                                             -4.76   2
c--     1       2       3       83.12           36.7    -4.64
c--     1       2       3       47.81           0.5     -4.73
c--                                                             -4.68   3
c--     1       3       3       48.59           35.0    -1.36
c--     1       3       3       19.53           0.5     -1.90
c--                                                             -1.63   4
c--     1               30      80.54           33.3    -4.72
c--     1               30      48.61           0.5     -4.81
c--                                                             -4.76   5
c--
c--     2       1       3       89.67           23.9    -6.58
c--     2       1       3       67.89           0.5     -6.74
c--                                                             -6.66   6
c--     2       2       3       80.28           23.3    -5.70
```

```
c--     2       2       3       57.73           0.5     -5.72
c--                                                              -5.71   7
c--     2       3       3       76.31           23.3    -5.30
c--     2       3       3       53.24           0.5     -5.27
c--                                                              -5.28   8
c--     2               30      79.80           23.3    -5.65
c--     2               30      59.51           0.5     -5.90
c--                                                              -5.78   9
c--     air                     69.14           21.1    -4.80
c--     air                     52.30           0.5     -5.18
c--                                                              -4.99   10
c--
c-------------------------------------------------------------------------
c-------------------------------------------------------------------------
c--
c--      Statistics are used to determine the allowable ranges and
c--        setpoints for control
c--
c--     statistics table for conductivity sensors
c--
c--      loc  array  mean   lower  upper   s.d.   s.e.  n   median   cv
c--      ~~~  ~~~~~  ~~~~   ~~~~~  ~~~~~   ~~~~   ~~~~  ~   ~~~~~~   ~~~
c--
c-- conductivity
c--
c--      pl1.1  12   2094   1259   3984    557    39  200   1972   26.6%
c--      pl1.2  13   5370   1410  24895   4543   321  200    361   84.6
c--      pl1.3  14   3000    431  10748   2025   143  200   2324   67.5
c--      pl2.1  15  22390  14905  28720   4400   311  200  23732   19.7
c--      pl2.2  16  11556   4256  20915   4394   310  200   9628   38.0
c--      pl2.3  17   3462   1285   7051   1127    79  200   3231   32.5
c--
c-- commercial conductivity
c--
c--      pl1.1  19   1428    904   1753    177    13  200   1456   12.4
c--      pl1.2  20    469    281   3613    478    33  200    385  102.1
c--      pl1.3  21    342    244    410     33     2  200    347    9.5
c--      pl2.1  22   1755    778   3046    712    50  200   1821   40.6
c--      pl2.2  23    508    100   1354    315    22  200    502   62.2
c--      pl2.3  24    335    262   1033    82      6  200    308   24.7
c--
c--       NOTE: BOUA is used to store the lower limit of reading
c--             BOUB is used to store the upper limit of reading
c-------------------------------------------------------------------------
c--
        DATA BOUA/-4.88,-4.825,-4.733,-1.911,-4.812,      !1-5
        +-6.742,-5.724,-5.273,-5.906,-5.189,              !5-10
        +1.0E+3,1259.,1410.,431.,14905.,                  !11-15
        +4256.,1285.,1.,904.,281.,                        !16-20
        +244.,778.,100.,262.,1.0E+3/                      !21-25
c--
c------------------------------------------------------------------
```

```
c--     boub is used for converting readings to valid data
c--     data values for BOUB array
        DATA BOUB/0.10,0.0969,0.0975,0.0823,0.0983,           !1-5
        +0.0931,0.0989,0.1018,0.0890,0.0818,                   !6-10
        +1.0,3984.,24895.,10748.,28720.,                       !11-15
        +20915.,7051.,1.,1753.,3613.,                          !16-20
        +410.,3046.,1354.,1033.,1.0/                           !21-25
c--
c----------------------------------------------------------------------
c--
c--     data array initialization
c--
        DATA BOUDAT/60*0./BDATA/60*0./BOUT/60*0./NARRAY/60/
c----------------------------------------------------------------------
c--
c--     Data and variables for conductivity and temperature readings
c--
c--     BCONL   low conductance test for conductance range
c--     BCONH   high conductance test for conductance range
c--     BRI     (input resistor) conductivity bridge range resistor
c--     BTMPRF  reference temperature for temperance correction
c--     BCONRF  reference values for range test
c--     BCONIN  conductance by temperature for temp correction
c--     BVOLTS  output rms voltage of conductivity bridge multivibrator
c--     BAI     --not used at present--set to 0.
c--     BTMPHZ  frequency(kHz) of reference resistor for temp mod check
c--             resistor gives reading of 56khz which equals 6 degrees c
c--
        DATA BCONL/1./BCONH/100./BCONRF/1.82,18.18,181.81/
        *BRI/1.0E+03,10.0E+03,100.0E+03/BVOLTS/1./BAI/00./
        *BTMPRF/25./BCONIN/0.02/BTMPHZ/6./
c--
c----------------------------------------------------------------------
c--
c--     Naming convention for arrays which contain switches or channels
c--     of sensors or pumps
c--
c--     M....   switch  or channels
c--     N....   number of switches
c--     .B...   array is used for boulder program
c--     ..TMP   temperature array
c--     ..CON   conductivity array
c--     ..PRS   commercial conductivity array
c--     ..PMP   pump array
c--     ..FFT   flow frequency
c--------------------------------------------------------------------------
c--
c--     Initialize arrays for temperature ,conductivity,and pumps
c--
c--     Channel number 9 is the location of the reference resistor
c--     This is the same as IBREFR the reference resistor is used
c--     to determine if a problem exists in a given module.
```

```
c.........................................................................
c--
c--     Conductivity    NBCON=number of conductivities
c--
        DATA MBCON/11,14,18,21,24,28,"000/NBCON/6/
c--
c.........................................................................
c--
c--     Temperature     NBTMP=number of temperatures
c--
        DATA MBTMP/38,10,13,16,17,20,23,26,27,30/NBTMP/10/
c--
c.........................................................................
c--
c--     Pump locations
c--
        DATA MBPMP/36,37/NBPMP/2/
c--
c.........................................................................
c--
c--     Commercial conductivity locations
c--
        DATA MBPRS/9,12,15,19,22,25,29/NBPRS/7/
c--
c.........................................................................
c--
c--     Ranges for conductivity
c--
        DATA MBCONR/2,3,4/NBCONR/3/
c--
c.........................................................................
c--
c--     Flow frequency locations
c--
        DATA MBFFT/34,35/NBFFT/2/
c--
c.........................................................................
c--
c--     Channel locations data for switches
c--
        DATA IBCONB/0/IBCOO/1/IBFCI/5/IBVFI/6/IBVFO/7/IBTCM/8/IBREFR/9/
        *IBWS/31/IBMPR/32/IBPPR/33/IBPNIC/39/
c--
c.........................................................................
c--
c--     Data for pump test DAC (digital analog convertor)
c--
        DATA BFTDAC/99.61,0.3906275,0.,0./
c--
c--     BFTDAC(1) base voltage for dac
c--     BFTDAC(2) factor to multiply reading by in binary conversion
c--     BFTDAC(3) ibit(7)  status of plot pump
```

```
c--     BFTDAC(4) ibit(8)  status of primary pump
c--
c--
c--     bftdac(2) is calculated by the folowing:
c--
c--     bftdac(2) = 99.61/255   where 255 is all the possible combinations
c--             for a 8 bit digital chip ((2**8)-1)
c--
c--     the above values are used for a binary conversion of the dac
c--     reading to determine which pumps are on and whic are off
c--
c--                     DAC output readings
c--                                                                    *
c--     pump status                    voltage   frequency     bit pattern
c--     ~~~~~~~~~~~~                    ~~~~~~   ~~~~~~~~~     ~~~~~~~~~~~
c--
c--     all off                          9.96       99.6        0000 0000
c--     manual plot 2 relay              9.81       98.1        0000 0100
c--     manual plot 1 relay              9.65       96.5        0000 1000
c--     manual plot pump                 9.34       93.4        0001 0000
c--     manual primary pump              8.71       87.1        0010 0000
c--     manual all on                    7.62       76.2        0011 1100
c--     computer plot pump               7.47       74.7        0100 0000
c--     computer primary pump            4.96       49.6        1000 0000
c--     computer primary and plot pump   2.46       24.6        1100 0000
c--     all on                           0.12        1.2        1111 1100
c--
c--     *NOTE: Bits 1 and 2 are not used and should always be zero
c--             ( bit 8 --> bit 1 )    8765 4321
c--
c-----------------------------------------------------------------------
c--
c--     Reference for channel locations
c--
c-----------------------------------------------------------------------
c--
c--     Variable     Channel     Comment
c--     ~~~~~~~~     ~~~~~~~     ~~~~~~~
c--     IBCONB          00       Conductance bridge
c--     IBCOO           01       Conductance oscillator output
c--     MBCONR(1)       02       Cond Range x 0.01
c--     MBCONR(2)       03       Cond Range x 0.1
c--     MBCONR(3)       04       Cond Range x 1.0
c--     IBFCI           05       Frequency counter input
c--     IBVFI           06       Voltage/frequency input
c--     IBVFO           07       Voltage/frequency output
c--     IBTCM           08       Temperature conversion module
c--     IBREFR          09       Reference resistor  5.46k
c--     MBTMP(1)        10       Temperature  Plot 1 station 1 3 cm
c--     MBCON(1)        11       Conductivity Plot 1 station 1
c--     MBPRS(1)        12       Comm Cond.      plot 1 station 1
c--     MBTMP(2)        13       Temperature  Plot 1 station 2 3 cm
```

```
c--     MBCON(2)        14      Conductivity Plot 1 station 2
c--     MBPRS(2)        15      Comm Cond.     plot 1 station 2
c--     MBTMP(3)        16      Temperature  Plot 1 station 3 3 cm
c--     MBTMP(4)        17      Temperature  Plot 1 station 3 30 cm
c--     MBCON(3)        18      Conductivity Plot 1 station 3
c--     MBPRS(3)        19      Comm Cond.     plot 1 station 3
c--     MBTMP(5)        20      Temperature  Plot 2 station 1 3 cm
c--     MBCON(4)        21      Conductivity Plot 2 station 1
c--     MBPRS(4)        22      Comm Cond.     plot 2 station 1
c--     MBTMP(6)        23      Temperature  Plot 2 station 2 3 cm
c--     MBCON(5)        24      Conductivity Plot 2 station 2
c--     MBPRS(5)        25      Comm Cond.     plot 2 station 2
c--     MBTMP(7)        26      Temperature  Plot 2 station 3 3 cm
c--     MBTMP(8)        27      Temperature  Plot 2 station 3 30 cm
c--     MBCON(6)        28      Conductivity Plot 2 station 3
c--     MBPRS(6)        29      Comm Cond.     plot 2 station 3
c--     MBTMP(6)        30      Ambient Air temperature
c--     IBWS            31      Wind speed
c--     IBMPR           32      Main pump relay (normally open drain)
c--     IBPPR           33      Plot pump relay (delay time before turning
c--                              on)
c--     MBFFT(1)        34      Plot 1 flow freq. transducer
c--                              (determines flow rate)
c--     MBFFT(2)        35      Plot 2 flow freq. transducer
c--                             (determines flow rate)
c--     MBPMP(1)        36      Plot 1 valve current relay
c--                              (turns on valve controller)
c--     MBPMP(2)        37      Plot 2 valve current relay
c--                              (turns on vave controller)
c--                     38      Reference resistor for temp check
c--                             (same as 09 but used to eliminate crossta)
c--     IBPNIC          39      flow test DAC converter
c-----------------------------------------------------------------------
c--             array locations for boudat array
c--             there are presently 60 values in this array
c-----------------------------------------------------------------------
c--
c--     BOUDAT(1)       reference temperature check
c--     BOUDAT(2)       temperature  plot 1 station 1 3 cm
c--     BOUDAT(3)       temperature  plot 1 station 2 3 cm
c--     BOUDAT(4)       temperature  plot 1 station 3 3 cm
c--     BOUDAT(5)       temperature  plot 1 station 3 30 cm
c--     BOUDAT(6)       temperature  plot 2 station 1 3 cm
c--     BOUDAT(7)       temperature  plot 2 station 2 3 cm
c--     BOUDAT(8)       temperature  plot 2 station 3 3 cm
c--     BOUDAT(9)       temperature  plot 2 station 3 30 cm
c--     BOUDAT(10)      ambient air temperature
c--     BOUDAT(11)      reference oscillator check
c--     BOUDAT(12)      conductivity plot 1 station 1
c--     BOUDAT(13)      conductivity plot 1 station 2
c--     BOUDAT(14)      conductivity plot 1 station 3
c--     BOUDAT(15)      conductivity plot 2 station 1
```

```
c--     BOUDAT(16)      conductivity plot 2 station 2
c--     BOUDAT(17)      conductivity plot 2 station 3
c--     BOUDAT(18)      reference comm cond. check
c--     BOUDAT(19)      comm cond.   plot 1 station 1
c--     BOUDAT(20)      comm cond.   plot 1 station 2
c--     BOUDAT(21)      comm cond.   plot 1 station 3
c--     BOUDAT(22)      comm cond.   plot 2 station 1
c--     BOUDAT(23)      comm cond.   plot 2 station 2
c--     BOUDAT(24)      comm cond.   plot 2 station 3
c--     BOUDAT(25)      wind speed
c--     BOUDAT(26)      main pump relay on (1)  off (0)
c--     BOUDAT(27)      DAC reading after pumps status corrected
c--     BOUDAT(28)      plot 1 flow rate
c--     BOUDAT(29)      plot 2 flow rate
c--     BOUDAT(30)      plot 1 valve relay on (1)  off (0)
c--     BOUDAT(31)      plot 2 valve relay on (1)  off (0)
c--     BOUDAT(32)      plot 1 time valve turned on
c--     BOUDAT(33)      plot 2 time valve turned on
c--     BOUDAT(34)      plot 1 date valve turned on
c--     BOUDAT(35)      plot 2 date valve turned on
c--     BOUDAT(36)      system time of previous read
c--     BOUDAT(37)      system date of previous read
c--     BOUDAT(38)      DAC reading coming into program (stability)
c--     BOUDAT(39)      temp reading ave - probes out of range plot 1
c--     BOUDAT(40)      temp reading ave - probes out of range plot 2
c--     BOUDAT(41)      cond reading ave - probes out of range plot 1
c--     BOUDAT(42)      cond reading ave - probes out of range plot 2
c--     BOUDAT(43)      comm cond read ave - probes out of range plot 1
c--     BOUDAT(44)      comm cond read ave - probes out of range plot 2
c--     BOUDAT(45)      average temperature plot 1
c--     BOUDAT(46)      average temperature plot 2
c--     BOUDAT(47)      average conductivity plot 1
c--     BOUDAT(48)      average conductivity plot 2
c--     BOUDAT(49)      average comm. cond. plot 1
c--     BOUDAT(50)      average comm. cond. plot 2
c--     BOUDAT(51)      total time on plot 1
c--     BOUDAT(52)      total time on plot 2
c--     BOUDAT(53)      acc flow plot 1 (in this period)
c--     BOUDAT(54)      acc flow plot 2 (in this period)
c--     BOUDAT(55)      most recent time relay on plot 1
c--     BOUDAT(56)      most recent time relay on plot 2
c--     BOUDAT(57)      most recent date relay on plot 1
c--     BOUDAT(58)      most recent date relay on plot 2
c--     BOUDAT(59)      date to write ave to disk
c--     BOUDAT(60)      counter for times thru always 1. used in acc file
c-------------------------------------------------------------
c--             bdata array locations
c--             array for daily and hourly averages
c-------------------------------------------------------------
c--
c--     BDATA(1)        reference temperature check
c--     BDATA(2)        temperature  plot 1 station 1 3 cm
```

```
c--     BDATA(3)        temperature  plot 1 station 2 3 cm
c--     BDATA(4)        temperature  plot 1 station 3 3 cm
c--     BDATA(5)        temperature  plot 1 station 3 30 cm
c--     BDATA(6)        temperature  plot 2 station 1 3 cm
c--     BDATA(7)        temperature  plot 2 station 2 3 cm
c--     BDATA(8)        temperature  plot 2 station 3 3 cm
c--     BDATA(9)        temperature  plot 2 station 3 30 cm
c--     BDATA(10)       ambient air temperature
c--     BDATA(11)       reference oscillator check
c--     BDATA(12)       conductivity plot 1 station 1
c--     BDATA(13)       conductivity plot 1 station 2
c--     BDATA(14)       conductivity plot 1 station 3
c--     BDATA(15)       conductivity plot 2 station 1
c--     BDATA(16)       conductivity plot 2 station 2
c--     BDATA(17)       conductivity plot 2 station 3
c--     BDATA(18)       reference comm cond. check
c--     BDATA(19)       comm cond.   plot 1 station 1
c--     BDATA(20)       comm cond.   plot 1 station 2
c--     BDATA(21)       comm cond.   plot 1 station 3
c--     BDATA(22)       comm cond.   plot 2 station 1
c--     BDATA(23)       comm cond.   plot 2 station 2
c--     BDATA(24)       comm cond.   plot 2 station 3
c--     BDATA(25)       wind speed
c--     BDATA(26)
c--     BDATA(27)
c--     BDATA(28)       plot 1 ave flow rate
c--     BDATA(29)       plot 2 ave flow rate
c--     BDATA(30)       plot 1 times valve relay on
c--     BDATA(31)       plot 2 times valve relay on
c--     BDATA(32)       plot 1 previous time valve on
c--     BDATA(33)       plot 2 previous time valve on
c--     BDATA(34)       plot 1 previous date valve on
c--     BDATA(35)       plot 2 previous date valve on
c--     BDATA(36)       system time of previous read
c--     BDATA(37)       system date of previous read
c--     BDATA(38)
c--     BDATA(39)       temp reading ave - probes out of range plot 1
c--     BDATA(40)       temp reading ave - probes out of range plot 2
c--     BDATA(41)       cond reading ave - probes out of range plot 1
c--     BDATA(42)       cond reading ave - probes out of range plot 2
c--     BDATA(43)       comm cond read ave - probes out of range plot 1
c--     BDATA(44)       comm cond read ave - probes out of range plot 2
c--     BDATA(45)       average temperature plot 1
c--     BDATA(46)       average temperature plot 2
c--     BDATA(47)       average conductivity plot 1
c--     BDATA(48)       average conductivity plot 2
c--     BDATA(49)       average comm. cond. plot 1
c--     BDATA(50)       average comm. cond. plot 2
c--     BDATA(51)       total time on plot 1
c--     BDATA(52)       total time on plot 2
c--     BDATA(53)       acc flow plot 1 ( this day)
c--     BDATA(54)       acc flow plot 2 ( this day)
```

```
c--     BDATA(55)       most recent time relay on plot 1
c--     BDATA(56)       most recent time relay on plot 2
c--     BDATA(57)       most recent date relay on plot 1
c--     BDATA(58)       most recent date relay on plot 2
c--     BDATA(59)       date writen to disk
c--     BDATA(60)       counter for times thru
c---------------------------------------------------------------------------
c--
c--
c--     SETPTC  conductivity set point for plots (1)plot 1  (2)plot2
c--     BCONAV  ave for cond for each plot (1)plot 1 (2)plot2
c--     BTMPAV  ave for tmp for each plot (1)plot 1 (2)plot2
c--     BPRSAV  ave for comm cond for each plot (1)plot 1 (2)plot2
c--     NPL     number of plots
c--     NPR     number of probes per plot
c--     NBT1    beginning array location for  plot 1 temperatures
c--     NBT2    beginning array location for  plot 2 temperatures
c--     NBC1    beginning array location for  plot 1 cond
c--     NBC2    beginning array location for  plot 2 cond
c--     NBP1    beginning array location for  plot 1 comm cond
c--     NBP2    beginning array location for  plot 2 comm cond
c--     LBCF    flag to see if cond in range (1)plot 1 (2)plot2
c--     LBTF    flag to see if tmperatures in range (1)plot 1 (2)plot2
c--     LBPF    flag to see if comm cond in range (1)plot 1 (2)plot2
c--     RANL    array location of cond for setpoint plot 1  (boumh2)
c--     RANH    array location of cond for setpoint plot 2  (boumh2)
c-------------------------------------------------------------------
        DATA BCONAV/2*0./BTMPAV/2*0./BPRSAV/2*0./NPL/2/NPR/3/
        *NBT1/2/NBT2/6/NBC1/12/NBC2/15/NBP1/19/NBP2/22/
        *LBCF/2*0/LBTF/2*0/LBPF/2*0/RANL/21./RANH/22./
        *SETPTC/11000.,1.0/
c-------------------------------------------------------------------------
c--
c--     other variables and set points
c--
c-----------------------------------------------------------------------
c--     WINDMX  maximum wind speed at which to operate the system
c--     TEMPMN  minimum temperature at which to operate the system
c--     LOCB    location used with data arrays to determine which
c--             variable in array is to be acted upon
c--     BAVETP  value to set temperatures if out of range
c--
c--     error flags for electronic module testing
c--
c--     KTMPF   checks temp module
c--     KCONF   checks conductivity module and oscillator
c--     KPRSF   checks conductivity module and oscillator
c--     KINTF   read error for data in bouint
c----------------------------------------------------------------------
        DATA WINDMX/30./TEMPMN/-5./LOCB/1/BAVETP/20.,20./
        DATA KTMPF/0/KCONF/0/KPRSF/0/KINTF/0/
c----------------------------------------------------------------------
```

```
c--
c--     KF..  --) FLAG          KS..   --) CURRENT STATE
c--
c--     for flags 1=on [turn pump on] 0=off [turn pump off]
c--
c--     KFBR    flag for relays  (1) plot1 (2) plot2
c--     KFBP    flag for pumps [main and plot pumps]
c--
c--     for current state 1=on [ pump on] 0=off [ pump off]
c--
c--     KSBR    current state of relays (1)plot1 (2)plot2
c--     KSBP    current state of pumps [main and plot pumps]
c--     KPUMP   array to store pump channels which need to be turned on
c--     KCNT    counter to see if any pumps need to be turned on
c--     KPC     counter to see if all pumps turned on
c--
c---------------------------------------------------------------------
        DATA KFBR/2*0/KFBP/0/KSBR/2*0/KSBP/0/
c---------------------------------------------------------------------
c--
c--     time to wait before rerunning program
c--     that is how often should prrogram be run
c--
c--     IBPOLH  wait in hours
c--     IBPOLM  wait in minutes
c--     IBPOLS  wait in seconds
c--     IBPOLT  wait in ticks
c--
c--     for practical purposes ticks and seconds are not realistic
c--     time intervals for boulder except for debugging.
c--     hours and minutes would be used for the polling period.
c---------------------------------------------------------------------
        DATA IBPOLH,IBPOLM,IBPOLS,IBPOLT/0,5,0,0,/
c---------------------------------------------------------------------
c--
c--     octal constants used for control sequences
c--
c---------------------------------------------------------------------
c--                      octal
c--     variable         value   meaning
c--     ~~~~~~~~         ~~~~~   ~~~~~~~
c--
c--     LA               "101    A
c--     LB               "102    B (ATTENTION CLEAR)
c--     LC               "103    C (INTERFACE CLEARCLOSE FOLLOWING CHANNEL)
c--     LR               "122    R (REMOTE SET)
c--     LDLE             "020    DLE (DATA LINK ESCAPE)
c--     LQU              "077    ? (UNLISTEN)
c--     LLAS             "051    ) (LISTEN ADDRESS SCANNER)
c--     LLACT            "175    } (LISTEN ADDRESS CONTROLLER)
c--     LTACT            "175    } (TALK ADDRESS CONTROLLER)
c--     LTAC             "112    J (TALK ADDRESS COUNTER)
```

```
c--     LLAC            "052    * (LISTEN ADDRESS COUNTER)
c--     LI              "111    I (INITIATE)
c--     LF              "106    F (FOR HANDSHAKING)
c--     LLF             "012    LINE FEED
c--     LCR             "015    CARRIGE RETURN
c-----------------------------------------------------------------------
        DATA LA/"101/LB/"102/LC/"103/LR/"122/LDLE/"020/LQU/"077/
        *LLAS/"051/LTACT/"175/LLACT/"175/LTAC/"112/LLAC/"052/
        *LI/"111/LF/"106/LCR/"015/LLF/"012/
c-----------------------------------------------------------------------
c--
c--     timing values for control sequences for ISLEEP system subroutine
c--     in RTLLI,RLLTI,RET
c--
c--     ITRH1   hours to wait
c--     ITRM1   minutes to wait
c--     ITRS1   seconds to wait
c--     ITRT1   ticks to wait
c--     ITRT2   ticks to wait
c--     ITRT3   ticks to wait
c--
c--     control sequences take from 2-15 ticks for minimum turn around
c--     depending upon the sequence sent
c-----------------------------------------------------------------------
        DATA ITRH1/0/ITRM1/0/ITRS1/0/ITRT1/2/ITRT2/7/ITRT3/15/
c-----------------------------------------------------------------------
c--
c--     active diagnostic assignments
c--
c--     byte locations for diagnostic data
c--
c=======================================================================
c-      VARIABLE       LOCATION  FUNCTION
c--     ~~~~~~~~       ~~~~~~~~  ~~~~~~~~
c--     JBDIO           "170     input/output device
c--     JBDPGM          "171     program execution byte
c--     JBDS1           "172     main subroutine
c--     JBDS2           "173     control or secondary subroutine
c--     JBDCH1          "174     channel to be acted upon
c--     JBDCH2          "175     override shutdown
c--     JBDCTL          "176     control subroutine loacation
c--     JBDERR          "177     error byte
c=======================================================================
c--     MAIN SUBROUTINE DATA ASSIGNMENTS FOR JBDS1
c--
c--     VARIABLE          VALUE           SUBROUTINE
c--     ~~~~~~~~          ~~~~~           ~~~~~~~~~~
c--     JBDAL             10              BOUDAL  (phone dialing)
c--      (+1)             11               passed timer o.k.
c--      (+2)             12               printed number o.k.
c--      (+3)             13               looks for carrier
c--      (+4)             14               test carrier
```

```
c--      (+5)           15                succesful attach
c--     JBTMP           20              BOUTMP  (temperature)
c--     JBWND           30              BOUWND  (wind speed)
c--     JBCON           40              BOUCON  (conductivity)
c--                     41              BOUOSC  (oscillator test)
c--     JBPRS           50              BOUPRS  (commercial conductivity)
c--     JBFFT           60              BOUFFT  (flow rate)
c--                     61              BOUDAC  (digital-analog convertor)
c--     JBERR           70              BOUER1  (error detection)
c--                     71              BOUER2  (shutdown)
c--     JBMAH           80              BOUMH1  (calcs ave cond.)
c--                     81              BOUMH2  (calcs ave readings)
c--                     82              BOUMHT  (calcs ave temp)
c--     JBPMP           90              BOUPMP  (manipulates pumps)
c--                     91              BOUPDC  (checks sync of pumps)
c--                     92              BOUPFT  (checks sync of relays)
c--                     93              BOUPFD  (turns pumps off if error)
c--     JBSTA           100             BOUSTA  (outputs current status)
c--     JBTIM           110             BOUTIM  (calcs time pumps on)
c--     JBDK1           120             BOUDK1  (outputs info to disk fil)
c--                     121             BOUDK2  (daily ave file operation)
c--     JBSLP           130             BOUSLP  (calcs time ref for sleep)
c--     JBPKC           140             BOUPKC  (reset diag bytes)
c--     JBINT           150             BOUIN1  (sets operating condition)
c--                     151             BOUIN2  (sets operating condition)
c--     JBFAL           250             BOUFAL  (exit program restart fro)
c==========================================================================
c--     SECONDARY SUBROUTINE DATA ASSIGNMENTS FOR JBDS2
c--
c--     VARIABLE        VALUE           SUBROUTINE
c--     ~~~~~~~~        ~~~~~           ~~~~~~~~~~
c--     JBRTL           10              BOURTL  (sets cond so data read)
c--     JBCHN           20              BOUCHN  (decimal channel to octal)
c--     JBRLL           30              BOURLL  (open/close scanner chann)
c--     JBWRT           40              BOUWRT  (reads information at sit)
c--     JBRET           50              BOURET  (channel command sent)
c--     JBRED           60              BOURED  (disk operations)
c--     JBBIT           70              BOUBIT  (calcs bit status from wo)
c=========================================================================
c--     miscellaneous data assignments
c--
c--     VARIABLE        VALUE   LOC.    FUNCTION
c--     ~~~~~~~~        ~~~~~   ~~~~~~  ~~~~~~~~
c--     JBDISK          0       JBDIO   output to disk
c--     JBTT            10      JBDIO   output to terminal
c--     JBECON          1       JBDERR  error oscillator
c--     JBEINT          2       JBDERR  error init disk read
c--     JBETMP          4       JBDERR  error conductance board
c--     JBEFFT          8       JBDERR  error dac
c--     JBEDAL          16      JBDERR  error phone
c--     JBEHND          32      JBDERR  error handshake
c--     JBEADD          64      JBDERR  error err on backup write
```

```
c--     JBESHT           128     JBDERR  error shut down
c--     JBOVR            1       JBDCH2  override shut down
c--     JBWAIT           10      JBGPGM  wait mode before starting
c--     JBRUN            20      JBDPGM  program running
c--     JBSLEP           30      JBDPGM  sleep period
c---------------------------------------------------------------------------
        DATA JBDIO/"170/JBDPGM/"171/JBDS1/"172/JBDS2/"173/
        *JBDCH1/"174/JBDCH2/"175/JBDCTL/"176/JBDERR/"177/
        DATA JBDAL/10/JBTMP/20/JBWND/30/JBCON/40/JBPRS/50/
        *JBFFT/60/JBERR/70/JBMAH/80/JBPMP/90/
        *JBSTA/100/JBTIM/110/JBDK1/120/JBSLP/130/JBPKC/140/
        *JBINT/150/JBFAL/250/
        DATA JBRTL/10/JBCHN/20/JBRLL/30/JBWRT/40/
        *JBRET/50/JBRED/60/JBBIT/70/
        DATA JBDISK/0/JBTT/10/JBECON/1/JBEINT/2/JBETMP/4/JBEFFT/8/
        *JBEDAL/16/JBEHND/32/JBEADD/64/JBESHT/128/JBOVR/1/JBWAIT/10/
        *JBRUN/20/JBSLEP/30/
c---------------------------------------------------------------------------
c--
c--     DIAGNOSTIC VALUES FOR CONTROL SUBROUTINE RTLLI, RLLTI, RET
c--     THESE ARE NAMED BOURTL.FOR, BOURLL.FOR, BOURET.FOR IN
c--     THE SYSTEM DIRECTORY
c--     THESE VALUES ARE IN LOCATION "176 AND ARE USED BY THE BACKGROUND
c--     PROGRAM. THEY ARE BYTE VALUES AND NOT WORDS.
c===========================================================================
c--
c--
c--     value           mnemonic        octal           handshake
c--     ~~~~~           ~~~~~~~~        ~~~~~           ~~~~~~~~~
c--
c--     SUBROUTINE RTLLI        REMOTE TALKER, LOCAL LISTENER INITIALIZATN
c--                             SETS CONDITIONS SO DATA CAN BE RECEIVED
c--
c--     REMOTE SET
c--     1                 DLE           020             na
c--     2                 R             122             na
c--
c--     ATTENTION SET
c--     3                 DLE           020             na
c--     4                 A             101             106
c--
c--     UNLISTEN
c--     5                 ?             077             077
c--
c--     TALK ADDRESS CONTROLLER
c--     6                 }             175             175
c--
c--     LISTEN ADDRESS COUNTER
c--     7                 *             052             052
c--
c--     ATTENTION CLEAR
c--     8                 DLE           020             na
```

```
c--     9                 B              102             na
c--
c--     SEND INITIATE
c--     10                I              111             111
c--
c--     ATTENTION SET
c--     11                DLE            020             na
c--     12                A              101             106
c--
c--     UNLISTEN
c--     13                ?              077             077
c--
c--     TALK ADDRESS COUNTER
c--     14                J              112             112
c--
c--     LISTEN ADDRESS CONTROLLER
c--     15                }              175             175
c--
c--     ATTENTION CLEAR
c--     16                DLE            020             na
c--     17                B              102             na
c--
c=========================================================================
c--     SUBROUTINE RLLTI        REMOTE LISTENER, LOCAL TALKER INITIAIZATIN
c--                             SETS CONDITIONS SO SCANNER CHANNELS CAN BE
c--                             OPENED AND CLOSED
c--
c--     INTERFACE CLEAR
c--     18                DLE            020             na
c--     19                C              103             106
c--
c--     REMOTE SET
c--     20                DLE            020             na
c--     21                R              122             na
c--
c--     ATTENTION SET
c--     22                DLE            020             na
c--     23                A              101             106
c--
c--     UNLISTEN
c--     24                ?              077             077
c--
c--     TALK ADDRESS CONTROLLER
c--     25                }              175             175
c--
c--     LISTEN ADDRESS SCANNER
c--     26                )              051             051
c--
c--     ATTENTION CLEAR
c--     27                DLE            020             na
c--     28                B              102             na
c--
```

```
c--     CLOSE FOLLOWING CHANNELS
c--     29                C              103              103
c=========================================================================
c--     SUBROUTINE RET          POKES A CARRIAGE RETURN AND LINE FEED
c--                             TELLS SCANNER THAT ALL COMMANDS HAVE BEENT
c--
c--     30                CR             015              015
c--
c-------------------------------------------------------------------------
        END
```

```
      SUBROUTINE BOUCHK
c +-------------------------------------------------------------------+
c | Program :    BOULD.REL                                             |
c | Module  :    BOUCHK.FOR                                            |
c | Function:    routine to output status information for debugging    |
c |                                                                   |
c +-------------------------------------------------------------------+
c--
c--   data common block
c--
      COMMON/BOULD/BOUDAT(60),BOUA(25),BOUB(25),BDATA(60),BOUT(60),
      *NARRAY,BCONAV(2),BTMPAV(2),BPRSAV(2),NPL,NPR,NBT1,NBT2,NBC1,
      *NBC2,NBP1,NBP2,LBCF(2),LBTF(2),LBPF(2),SETPTC(2),LOCB,
      *KFBR(2),KFBP,KSBR(2),KSBP,KTMPF,KCONF,KPRSF,KINTF,KWNDF,KFFTF,
      *KPUMP(2)
c--
c--   channel switching common block
c--
      COMMON/CHAN/IBCONB,IBCOO,IBFCI,IBVFI,IBVFO,IBTCM,IBREFR,
      *IBWS,IBMPR,IBPPR,IBPNIC,MBCON(7),NBCON,MBTMP(10),
      *NBTMP,MBPMP(2),NBPMP,MBPRS(7),NBPRS,MBFFT(2),NBFFT,
      *MBCONR(3),NBCONR
c--
c--   data checking common block
c--
      COMMON/CONDCL/WINDMX,TEMPMN,BCONL,BCONH,BCONRF(3),BRI(3),
      *BVOLTS,BAI,BTMPRF,BCONIN,BAVETP(2),BTMPHZ,RANL,RANH,BFTDAC(4)
c--
c--   control character common block
c--
      COMMON /CONTRL/LA,LB,LC,LR,LDLE,LQU,LLAS,LTACT,LLACT,LTAC,
      *LLAC,LI,LF,LCR,LLF,ITRH1,ITRM1,ITRS1,ITRT1,ITRT2,ITRT3
c--
c--   diagnostic block
c--
      LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
      INTEGER*4 I1AREA,I2AREA
      COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
      *LTERM,ITERM,ATIMES,BTIMES,IADPOK,IADPEK,IPHONE,ILDTR,IHDTR,
      *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
      *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
      *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
      *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
      *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
      *JBOVR,JBWAIT,JBSLEP
c--
c--   file information block
c--
      COMMON/FILE/BOUDEV,MABLK(4),MBBLK(4),MCBLK(4),MDBLK(4),
      *MEBLK(4),MFBLK(4),MGBLK(4).MHBLK(4),MIBLK(4),MJBLK(4),MKBLK(4),
      *MLBLK(4),MMBLK(4),MNBLK(4),MOBLK(4),MPBLK(4),MQBLK(4),MRBLK(4),
      *MSBLK(4),
          *MABLKL,MABLKW,MBBLKL,MBBLKW,MCBLKL,
          *MCBLKW,MDBLKL,MDBLKW,MEBLKL,MEBLKW,MFBLKL,MFBLKW,MGBLKL,
          *MGBLKW,MHBLKL,MHBLKW,MIBLKL,MIBLKW,MJBLKL,MJBLKW,MKBLKL,
          *MKBLKW,MLBLKL,MLBLKW,MMBLKL,MMBLKW,MNBLKL,MNBLKW,MOBLKL,
          *MOBLKW,MPBLKL,MPBLKW,MQBLKL,MQBLKW,MRBLKL,MRBLKW,MSBLKL,MSBLKW
c--
c--       telephone number common block
c--
          LOGICAL*1 NUMBOU,NUMADA
          COMMON/NUMBER/NUMBOU(12),NUMADA(9)
c--
c--       base address common block
c--
          COMMON/PROCES/IAD103,IAD123,ITM03,ITM23
c--
c--       timer failure common block
c--
          COMMON/TIMER/ITMH,ITMM,ITMS,ITMT,ITDH,ITDM,ITDS,ITST
c--
c--       main program sleep block
c--
          COMMON/WAIT/IBPOLH,IBPOLM,IBPOLS,IBPOLT,IBCC,IBCH,IBCM,
          *IBCS,IBCT
c--
c--       ()()()()()()()()()()()()()()()()()()()()()()()()()()()()()()()()
c--
c--       formfeed to top of page
8         WRITE (LTERM,55) "014
c--
c--       list hierarchy
c--
          WRITE (LTERM,52)
52        FORMAT (//,' THE HIERARCHY OF SUBROUTINES AT '/
          1' THE TIME OF PRINT ARE: ')
          CALL PUTSTR (LTERM,LEVEL1)
          CALL PUTSTR (LTERM,LEVEL2)
          CALL PUTSTR (LTERM,LEVEL3)
c--
c--       list boudat array
c--
          WRITE (LTERM,19)
          WRITE (LTERM,20) (L, BOUDAT(L),L=1,60)
c--
c--       list averages and setpoints
c--
          WRITE (LTERM,30) (L,BCONAV(L),L,BTMPAV(L),L,BPRSAV(L),L=1,2)
          WRITE (LTERM,40) (SETPTC(I),I=1,2)
c--
c--       list flags and pump staus
c--
          WRITE (LTERM,50) (LBCF(L),LBTF(L),LBPF(L),L=1,2)
          WRITE (LTERM,60) ((KFBR(L),L=1,2),KFBP)
```

```
      WRITE (LTERM,65) ((KSBR(L),L=1,2),KSBP)
      WRITE (LTERM,70) KTMPF,KCONF,KPRSF,KINTF,KWNDF,KFFTF
      WRITE (LTERM,80) NARRAY,NPL,NPR,NBT1,NBT2,NBC1,NBC2,NBP1,NBP2
c--
      WRITE (LTERM,55) "014
      RETURN
c-----------------------------------------------------------------------
c--
c--   format statements
c--
c----------------------------------------------------------------------
19    FORMAT (' BOUDAT ARRAY')
20    FORMAT (5(X,I3,'==> ',F10.2))
55    FORMAT (1A1)
60    FORMAT (' FLAG => PLOT 12,MAIN ',3I5)
65    FORMAT (' CURR STAT => PLOT 12,MAIN ',3I5)
30    FORMAT (' BCONAV',I1,F10.2,'     BTMPAV',I1,
     *F10.2,'         BPRSAV',I1,,F10.2)
40    FORMAT (' SETPTC ',2F10.2)
50    FORMAT(' LBCF',I5,'      LBTF',I5,'       LBPF',I5)
70    FORMAT (' KTMPF',I2,5X,'KCONF',I2,5X,'KPRSF',I2,5X,'KINTF',I2,5X,
     *'KWNDF',I2,5X,'KFFTF',I2)
80    FORMAT ('NARRAY',I4,X,'NPL',I4,X,'NPR',I4,X,'NBT1',I4,X,
     *'NBT2',I4,X,'NBC1',I4,X,'NBC2',I4,X,'NBP1',I4,X,'NBP2',I4,)
      END
      SUBROUTINE BOUCON
c +------------------------------------------------------------------+
c | Program :    BOULD.REL                                           |
c | Module  :    BOUCON.FOR                                          |
c | Function:    Reads homemade commercial conductivity probes       |
c |                                                                  |
c +------------------------------------------------------------------+
c--
c--   data common block
c--
      COMMON/BOULD/BOUDAT(60),BOUA(25),BOUB(25),BDATA(60),BOUT(60),
     *NARRAY,BCONAV(2),BTMPAV(2),BPRSAV(2),NPL,NPR,NBT1,NBT2,NBC1,
     *NBC2,NBP1,NBP2,LBCF(2),LBTF(2),LBPF(2),SETPTC(2),LOCB,
     *KFBR(2),KFBP,KSBR(2),KSBP,KTMPF,KCONF,KPRSF,KINTF,KWNDF,KFFTF,
     *KPUMP(2)
c--
c--   channel switching common block
c--
      COMMON/CHAN/IBCONB,IBCDO,IBFCI,IBVFI,IBVFO,IBTCM,IBREFR,
     *IBWS,IBMPR,IBPPR,IBPNIC,MBCON(7),NBCON,MBTMP(10),
     *NBTMP,MBPMP(2),NBPMP,MBPRS(7),NBPRS,MBFFT(2),NBFFT,
     *MBCONR(3),NBCONR
c--
c--   data checking common block
c--
      COMMON/CONDCL/WINDMX,TEMPMN,BCONL,BCONH,BCONRF(3),BRI(3),
     *BVOLTS,BAI,BTMPRF,BCONIN,BAVETP(2),BTMPHZ,RANL,RANH,BFTDAC(4)
c--
c--   control character common block
c--
      COMMON /CONTRL/LA,LB,LC,LR,LDLE,LQU,LLAS,LTACT,LLACT,LTAC,
     *LLAC,LI,LF,LCR,LLF,ITRH1,ITRM1,ITRS1,ITRT1,ITRT2,ITRT3
c--
c--   diagnostic block
c--
      LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
      INTEGER*4 I1AREA,I2AREA
      COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
     *LTERM,ITERM,ATIMES,BTIMES,IADPOK,IADPEK,IPHONE,ILDTR,IHDTR,
     *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
     *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
     *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
     *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
     *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
     *JBOVR,JBWAIT,JBSLEP
c--
c--   ----------------------------------------------------------------
c--
c--
c--   bg program diagnostics
c--
      CALL IPOKEB(JBDS1,JBCON)
```

```
c--
c--     timing failure traceback
c--
        CALL SCOPY (' BOUCON',LEVEL2)
        CALL SCOPY ('      ',LEVEL3)
c--
c--     temporary diagnostics for debugging
c--
D       CALL PUTSTR(LTERM,' LOCATION: ','0')
D       CALL PUTSTR(LTERM,LEVEL1,' ')
D       CALL PUTSTR(LTERM,LEVEL2,' ')
D       CALL PUTSTR(LTERM,LEVEL3,' ')
c--
c----------------------------------------------------------------------
c--
c--             channel switching section stats here
c--             see boudat.for for channel assignments
c--
c----------------------------------------------------------------------
c--
D       WRITE(LTERM,1500) (MBCONR(KL),KL=1,3)
D1500   FORMAT (X,'BOUCON MBCONR ',3(2X,I5))
        DO 30,L=1,NBCON
c--
c--     determine which ave temp to use
c--
        IF (L .GE.1.AND. L .LE.3) BTMP=BTMPAV(1)        !ave for plot 1
        IF (L .GE.4) BTMP=BTMPAV(2)     !ave for plot 2
c--
c--     locb is calculated from the present location
c--     plus the number of temperatures read
c--     +1 for oscillator location
c--
        LOCB=(L+1)+NBTMP        !array locator for boudat array
        LTEST=1 !conductivity range reference location
c--
20      CALL RLLTI(MFLAG)       !sets conditions so channels can be closed
        IF (MFLAG.EQ.1) GO TO 20
c--
c--     close voltage to frequency convertor ouput and test error flag
c--
        CALL BOUCHN (IBVFO,MFLAG)
        IF (MFLAG.EQ.1) GO TO 20
c--
c--     close frequency counter input
c--
        CALL BOUCHN (IBFCI,MFLAG)
        IF (MFLAG.EQ.1) GO TO 20
c--
c--     close voltage to frequency input
c--
        CALL BOUCHN (IBVFI,MFLAG)
```

```
      IF (MFLAG.EQ.1) GO TO 20
c--
c--   close conductance bridge
c--
      CALL BOUCHN (IBCONB,MFLAG)
      IF (MFLAG.EQ.1) GO TO 20
c--
c--   close specified channel
c--
      CALL BOUCHN (MBCON(L),MFLAG)
      IF (MFLAG.EQ.1) GO TO 20
c--
c--   close range selector
c--
      CALL BOUCHN (MBCONR(LTEST),MFLAG)
      IF (MFLAG.EQ.1) GO TO 20
c--
      CALL RET(MFLAG) !tells scanner all commands have been sent
      IF (MFLAG.EQ.1) GO TO 20
c--
      CALL RTLLI(MFLAG)       !sets conditions so data can be received
      IF (MFLAG.EQ.1) GO TO 20
c--
      CALL WRITE(BOUDAT(LOCB),MFLAG)  !converts data to real*4 number
      IF (MFLAG.EQ.1) GO TO 20
c--
D     WRITE (LTERM,9000) BOUDAT(LOCB),BCONRF(LTEST),BCONL,BCONH,LOCB,
D     *LTEST,LEVEL2
D9000 FORMAT (X,5X,'READING',E12.5,5X,'REF',E12.5,5X,'RANGE',2E12.5,
D     */,' LOC',I5,5X,'LTEST ',I5,10A1)
c--
      LTEST=LTEST+1   !decrement reference location channel
      IF (LTEST.GT.4) GO TO 40        !error only three ranges
c--
c--   is reading within proper range
c--
      IF (BOUDAT(LOCB).GT.BCONH.XOR.BOUDAT(LOCB).LT.BCONL) GO TO 20
      LTEST=LTEST-1   !decrement ltest back to proper range
c--
c--
c--   do conductivity calculations
c--    1 volt = 10 hz
c--     divide by 10 to get reading back into volts
c--   conductivity= comductivity+ teperature correction
c--     temp correction =[(Ref.temp.) - ((ave temp)*cond.correction)*10]
c--          *1000 to convert to micro/Siemens
c--
31    IF (BOUDAT(LOCB).EQ.0.0) GOTO 333       !prevent 0. divide
333   TMP=BOUDAT(LOCB)/(10.*BRI(LTEST)*BVOLTS)
      BOUDAT(LOCB)=TMP*(1+(BTMPRF-BTMP)*BCONIN)*10.E+06
c--
D     WRITE (LTERM,*) BOUDAT(LOCB),LOCB,LTEST,BCONRF(LTEST)
```

```
      GOTO 30
40    WRITE (LTERM,41) L
41    FORMAT (/////' PROBLEM IN CONDUCTIVITY RANGE. CHANNEL ',I5,////)
30    CONTINUE
      RETURN
      END
      SUBROUTINE BOUDAC
c +--------------------------------------------------------------------+
c | Program :    BOULD.REL                                              |
c | Module  :    BOUDAC.FOR                                             |
c | Function:    Reads Digital to Analog converter                      |
c |              This gives an indication on how stable the dac and relay|
c |              in the switches are. This value is compared to that in |
c |              BOUPDC.FOR  for the stability                          |
c |                                                                     |
c |               Bit codes are explained in BOUDAT.FOR                 |
c |                                                                     |
c +--------------------------------------------------------------------+
c--
c--     data common block
c--
      COMMON/BOULD/BOUDAT(60),BOUA(25),BOUB(25),BDATA(60),BOUT(60),
     *NARRAY,BCONAV(2),BTMPAV(2),BPRSAV(2),NPL,NPR,NBT1,NBT2,NBC1,
     *NBC2,NBP1,NBP2,LBCF(2),LBTF(2),LBPF(2),SETPTC(2),LOCB,
     *KFBR(2),KFBP,KSBR(2),KSBP,KTMPF,KCONF,KPRSF,KINTF,KWNDF,KFFTF,
     *KPUMP(2)
c--
c--     channel switching common block
c--
      COMMON/CHAN/IBCONB,IBCOO,IBFCI,IBVFI,IBVFO,IBTCM,IBREFR,
     *IBWS,IBMPR,IBPPR,IBPNIC,MBCON(7),NBCON,MBTMP(10),
     *NBTMP,MBPMP(2),NBPMP,MBPRS(7),NBPRS,MBFFT(2),NBFFT,
     *MBCONR(3),NBCONR
c--
c--     data checking common block
c--
      COMMON/CONDCL/WINDMX,TEMPMN,BCONL,BCONH,BCONRF(3),BRI(3),
     *BVOLTS,BAI,BTMPRF,BCONIN,BAVETP(2),BTMPHZ,RANL,RANH,BFTDAC(4)
c--
c--     control character common block
c--
      COMMON /CONTRL/LA,LB,LC,LR,LDLE,LQU,LLAS,LTACT,LLACT,LTAC,
     *LLAC,LI,LF,LCR,LLF,ITRH1,ITRM1,ITRS1,ITRT1,ITRT2,ITRT3
c--
c--     diagnostic block
c--
      LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
      INTEGER*4 I1AREA,I2AREA
      COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
     *LTERM,ITERM,ATIMES,BTIMES,IADPOK,IADPEK,IPHONE,ILDTR,IHDTR,
     *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
     *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
     *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
     *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
     *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
     *JBOVR,JBWAIT,JBSLEP
c----------------------------------------------------------------------
c--
```

```
c--
c--     bits common
c--
        COMMON/BITS/IBIT(16),IWORD
c--
        REAL*4 LABEL(4,3)
        DATA LABEL/'Plot','Plot','Plot','Prim',
        *' 1 r',' 2 r',' pum','ary ',
        *'elay','elay','p   ','pump'/
c--
c--     diagnostic information
c--
c--     bg diagnostics
c--
        CALL IPOKEB(JBDS1,JBFFT+1)
c--
c--     timing failure traceback diagnostics
c--
        CALL SCOPY (' BOUDAC',LEVEL2)
        CALL SCOPY ('       ',LEVEL3)
c--
c--     temporary diagnostics
c--
D       CALL PUTSTR(LTERM,' LOCATION: ','0')
D       CALL PUTSTR(LTERM,LEVEL1,' ')
D       CALL PUTSTR(LTERM,LEVEL2,' ')
D       CALL PUTSTR(LTERM,LEVEL3,' ')
c--
c-----------------------------------------------------------------------
c--
c--
c--
19      ITRY=0  !init zero check
        LOCB=38 !array location for boudat loc (38)
c--
c--     start control sequence
c--
20      CALL RLLTI(MFLAG)       !set conditions so channels can be closed
        IF (MFLAG.EQ.1) GO TO 20
c--
c--     close frequency counter input
c--
        CALL BOUCHN (IBFCI,MFLAG)
        IF (MFLAG.EQ.1) GO TO 20
c--
c--     close voltage frequency input
c--
        CALL BOUCHN (IBVFI,MFLAG)
        IF (MFLAG.EQ.1) GO TO 20
c--
c--     close voltage/frequency output
c--
```

```
      CALL BOUCHN (IBVFO,MFLAG)
      IF (MFLAG.EQ.1) GO TO 20
c--
c--   close DAC converter
c--
      CALL BOUCHN (IBPNIC,MFLAG)
      IF (MFLAG.EQ.1) GO TO 20
c--
c--   finish control sequence
c--
      CALL RET(MFLAG) !tells scanner all commands have been sent
      IF (MFLAG.EQ.1) GO TO 20
c--
      CALL RTLLI(MFLAG)        !sets conditions so data can be received
      IF (MFLAG.EQ.1) GO TO 20
c--
      CALL WRITE(BOUDAT(LOCB),MFLAG)  !converts data to real*4 number
      IF (MFLAG.EQ.1) GO TO 20
c--
c--   check for zero condition
c--   a zero may not be a zero but a bad reading ckeck and see
c--
      ITRY=ITRY+1
      IF (ITRY.LE.5 .AND.BOUDAT(LOCB) .EQ. 0.) GOTO 20
c--
c--
c--   check to see if in proper range
c--   if not go back and get reading again
c--
      IF (BOUDAT(LOCB) .LT. 0.0
      * .OR. BOUDAT(LOCB) .GT.100.0)
      *   GOTO 19
      IF (BOUDAT(LOCB) .EQ. 0.0) GOTO 999
c--
c--   convert reading
c--
      CALC = (BFTDAC(1)-BOUDAT(LOCB))/BFTDAC(2)
      IWORD=CALC
      ZHOLD=DIM(CALC,FLOAT(IWORD))
      IF (ZHOLD.GE.0.5) IWORD=IWORD+1
C
      CALL BOUBIT
c--
c--   store current status of pumps for test in boufft
      BFTDAC(3)=IBIT(7)
      BFTDAC(4)=IBIT(8)
c--
c--   out put results for debug
c--
D     WRITE(LTERM,9000) BOUDAT(LOCB),IWORD,SYSTM1,SYSDAT,LEVEL2
c--
c--   check to see if any problems with the readings
```

```
c--
c--     is pump status correct
c--
        IF (KFBP .EQ. IBIT(8)
        * .AND. KFBP .EQ. IBIT(7))
        *   GOTO 900
c--
c--     pumps out of sync
c--
        WRITE (LTERM,*) 'BOUDAC - Pumps out of sync'
        WRITE (LTERM,800) KFBP,(IBIT(K),K=8,1,-1),BOUDAT(LOCB)
c--
c--     section to tell whic pumps out of sync
c--
c--     was primary pump the one out of sync????
c--
36      IF (IBIT(8) .EQ. KFBP) GOTO 35
        WRITE (LTERM,*) 'BOUDAC - Primary pump out of sync.'
c--
c--     was plot pump the one out of sync?????
c--
35      IF (IBIT(7) .EQ. KFBP) GOTO 900
        WRITE (LTERM,*) 'BOUDAC - Plot pump out of sync.'
c--
c--     check to see if any manual operation!!
c--
900     DO 901 I=3,6
901     IF (IBIT(I).EQ.1) GOTO 905
        GOTO 30
c--
c--     some pumps on tell which ones
c--
905     WRITE (LTERM,*) 'BOUDAC - Manual operation status'
        DO 907 I=1,4
907     IF (IBIT(I+2).EQ.1) WRITE (LTERM,908) (LABEL(I,J),J=1,3)
c--
c--     normal return
c--
30      RETURN
999     WRITE (LTERM,*) 'BOUDAC - DAC error',BOUDAT(LOCB)
        WRITE (LTERM,*) 'BOUDAC - DAC should not be zero.'
        RETURN
c+-----------------------------------------------------------------+
c|                                                                 |
c|                      format statements                          |
c|                                                                 |
c+-----------------------------------------------------------------+
800     FORMAT (X,'BOUDAC Pump flag',I3,3X,'BITS',2X,8I2,
        *3X,'DAC',F5.2)
908     FORMAT (X,3A4)
D9000   FORMAT (X,5X,'DAC',E12.5,5X,I5,F10.0,5X,F10.0,5X,10A1)
        END
        SUBROUTINE DIALT(NUMB)
c +----------------------------------------------------------------------+
c | Program :    BOULD.REL                                               |
c | Module  :    BOUDAL.FOR                                              |
c | Function:    PHONE DIALLING ROUTINE                                  |
c |                phone number is in variable NUMB                      |
c |                phone number is assigned in main driver               |
c |                                                                      |
c +----------------------------------------------------------------------+
c--
c--     diagnostic block
c--
        LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
        INTEGER*4 I1AREA,I2AREA
        COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
        *LTERM, ITERM, ATIMES, BTIMES, IADPOK, IADPEK, IPHONE, ILDTR, IHDTR,
        *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
        *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
        *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
        *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
        *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
        *JBOVR,JBWAIT,JBSLEP
c--
c--     timer failure common block
c--
        COMMON/TIMER/ITMH,ITMM,ITMS,ITMT,ITDH,ITDM,ITDS,ITST
c--
c--************************************************************
c--
        EXTERNAL FAIL
c--
c--     set bit configuration and counters
c--
        DATA ITOGLE/"167777/ICNT1,ICNT2,ICNT3/0,0,0/
c--
c--     tell bg where you are
c--
        CALL IPOKEB (JBDS1,JBDAL)       !passed timer o.k.
c--
c--     timing failure diagnostics
c--
        CALL SCOPY (' DIAL',LEVEL2)
        CALL SCOPY ('     ',LEVEL3)
c--
c--     temporary diagnostics for debugging
c--
D       CALL PUTSTR(LTERM,' LOCATION: ','0')
D       CALL PUTSTR(LTERM,LEVEL1,' ')
D       CALL PUTSTR(LTERM,LEVEL2,' ')
D       CALL PUTSTR(LTERM,LEVEL3,' ')
c--
```

```
c--
c--     following  line set and test timer to make sure there
c--     is a normal return. Timer is set for 5 minutes.
c--     if communications with greenhouse takes longer than 5
c--     minutes subroutine FAIL is called. Normal communication time
c--     is about 0.5 minutes.
c--
        IF( ITIMER (ITDH,ITDM,ITDS,ITDT,I2AREA,3,FAIL).NE.0)
        * WRITE (LTERM,334)
c--
        CALL IPOKEB (JBDS1,JBDAL+1)      !passed timer o.k.
c--
55      CALL MTATCH (ITERM)      !attatch remote terminal
c--
        CALL IPOKE (IPHONE,ILDTR)        !toggle dtr low
        CALL ISLEEP (0,0,2,0)            !time to disconnect dtr
c--
c--     start the dialing  and carrier detect sequence
c--
50      CALL IPOKE (IPHONE,IHDTR)        !toggle dtr hi
        CALL ISLEEP (0,0,2,0)    !wait for dtr to be set
c--
        CALL MTPRNT (ITERM,NUMB)         !call number
c--
        CALL IPOKEB (JBDS1,JBDAL+2)      !printed number o.k.
        CALL ISLEEP (0,0,15,0)           !give time to dial
c--
100     ICARR=(.NOT. (ITOGLE .OR. IPEEK (IPHONE)))      !carrier ???
        CALL IPOKEB (JBDS1,JBDAL+3)      !tell bg where you are
        IF (ICARR .EQ. 0) GO TO 300      !look for carrier
        CALL IPOKEB (JBDS1,JBDAL+4)      !test carrier
c--
        CALL ISLEEP (0,0,5,0)            !wait and try again
c--
        ICNT1 = ICNT1 + 1        !inc carrier look counter
        IF (ICNT1 .LE. 10) GO TO 100     !go back and look for carrier
c--
        CALL IPOKE (IPHONE,ILDTR)        !carrier not detected toggle dtr w
        IPOK=(IPEEKB(JBDERR))+JBEDAL     !get where you are
        CALL IPOKEB(JBDERR,IPOK)         !tell bg what's happening
        CALL ISLEEP (0,0,5,0)    !give time to reset dtr
c--
        ICNT1 = 0        !init carrier look counter
        ICNT2 = ICNT2 + 1        !inc retry counter
        IF (ICNT2 .LE. 10) GO TO 50      !retry dialing
c--
        WRITE (LTERM,200)        !tell unsucessful
        ICNT3=ICNT3+1    !increment failure counter
        ICNT2=0 !init retry counter
        CALL ISLEEP (0,0,10,0)  !wait 10 seconds and try again
c--
c--
```

```
        IF (ICNT3.GT.3) CALL FAIL(3) !go to fail and restart - phone probm
        GO TO 55
300     CONTINUE
c--
        CALL IPOKEB (JBDS1,JBDAL+5)      !succesful attach
c--
        CALL MTDTCH (ITERM)      !detatch remote terminal
c--
        CALL ICMKT(3,I2AREA)     !cancels timer
c--
        RETURN
c--
c--     format statements
c--
200     FORMAT (' 10 UNSUCCESSFUL ATTEMPTS TO ATTATCH THE TELEPHONE')
334     FORMAT (' DIAL TIMER DID NOT HAVE NORMAL RETURN')
        END
        SUBROUTINE BOUDK1
c +----------------------------------------------------------------------+
c | Program :    BOULD.REL                                               |
c | Module  :    BOUDK1.FOR                                              |
c | Function:    WRITES DATA TO STSTUS AND ACCUMULATOR FILES             |
c |                                                                      |
c +----------------------------------------------------------------------+
c--
c--     data common block
c--
        COMMON/BOULD/BOUDAT(60),BOUA(25),BOUB(25),BDATA(60),BOUT(60),
        *NARRAY,BCONAV(2),BTMPAV(2),BPRSAV(2),NPL,NPR,NBT1,NBT2,NBC1,
        *NBC2,NBP1,NBP2,LBCF(2),LBTF(2),LBPF(2),SETPTC(2),LOCB,
        *KFBR(2),KFBP,KSBR(2),KSBP,KTMPF,KCONF,KPRSF,KINTF,KWNDF,KFFTF,
        *KPUMP(2)
c--
c--     diagnostic block
c--
        LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
        INTEGER*4 I1AREA,I2AREA
        COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
        *LTERM, ITERM, ATIMES,BTIMES, IADPOK, IADPEK, IPHONE, ILDTR, IHDTR,
        *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
        *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
        *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
        *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
        *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
        *JBOVR,JBWAIT,JBSLEP
c--
c--     file information block
c--
        COMMON/FILE/BOUDEV,MABLK(4),MBBLK(4),MCBLK(4),MDBLK(4),
        *MEBLK(4),MFBLK(4),MGBLK(4),MHBLK(4),MIBLK(4),MJBLK(4),MKBLK(4),
        *MLBLK(4),MMBLK(4),MNBLK(4),MOBLK(4),MPBLK(4),MQBLK(4),MRBLK(4),
        *MSBLK(4),
        *MABLKL,MABLKW,MBBLKL,MBBLKW,MCBLKL,
        *MCBLKW,MDBLKL,MDBLKW,MEBLKL,MEBLKW,MFBLKL,MFBLKW,MGBLKL,
        *MGBLKW,MHBLKL,MHBLKW,MIBLKL,MIBLKW,MJBLKL,MJBLKW,MKBLKL,
        *MKBLKW,MLBLKL,MLBLKW,MMBLKL,MMBLKW,MNBLKL,MNBLKW,MOBLKL,
        *MOBLKW,MPBLKL,MPBLKW,MQBLKL,MQBLKW,MRBLKL,MRBLKW,MSBLKL,MSBLKW
c-----------------------------------------------------------------------
c--
c--     bg diagnostic traceback
c--
        CALL IPOKEB (JBDS1,JBDK1)
c--
c--     timing failure traceback
c--
        CALL SCOPY (' BOUDK1     ',LEVEL2)
c--
c--     tempory diagnostics for debugging
c--
```

```
D        CALL PUTSTR(LTERM,' LOCATION: ','0')
D        CALL PUTSTR(LTERM,LEVEL1,' ')
D        CALL PUTSTR(LTERM,LEVEL2,' ')
D        CALL PUTSTR(LTERM,LEVEL3,' ')
c--
c---------------------------------------------------------------------
c--      write to disk file
c---------------------------------------------------------------------
c--
c--      write data for current readings
c--
         IF (BOUDAT(59).EQ.0.) BOUDAT(59)=SYSDAT
         CALL BOURED (MABLK,BOUDAT,MABLKW/2,MABLKW,0,'WR','BO',IERR)
         IF (IERR.NE.0) GOTO 1005
c--
c--      write current data to backup file
c--
80       CALL BOURED (MMBLK,BOUDAT,MMBLKW/2,MMBLKW,0,'WR','BO',IERR)
         IF (IERR.NE.0) GOTO 1006
c--
c--      initialize temporary array
c--
         DO 30 I=1,NARRAY
30       BOUT(I)=0.
c--
c--      read accumulative file from disk
c--
81       CALL BOURED(MBBLK,BOUT,MBBLKW/2,MBBLKW,0,'RE','OP',LERR)
c--
c--      check for errors
c--
         IF (LERR.NE.0) GOTO 1000
c--
c--
c--      add to present readings
c--
c--      acc plot readings
c--
c--
         DO 31, I=1,25
31       BOUT(I)=BOUT(I)+BOUDAT(I)
c--
c--      times relay on
c--
         DO 33, I=30,31
33       BOUT(I)=BOUT(I)+BOUDAT(I)
c--
c--      dates and times of last readings
c--
         DO 32, I=32,37
32       BOUT(I)=BOUDAT(I)
c--
```

```
c--     averages and flags
c--
        DO 35, I=39,54
35      BOUT(I)=BOUT(I)+BOUDAT(I)
c--
c--     disk write and acc counter
c--
        DO 34, I=55,NARRAY-1
34      BOUT(I)=BOUDAT(I)
        BOUT(NARRAY)=BOUT(NARRAY)+BOUDAT(NARRAY)
c--
c--     write data,close file and check for errors
c--
        CALL BOURED(MBBLK,BOUT,MBBLKW/2,MBBLKW,0,'WR','CL',LERR)
        IF (LERR.NE.0) GOTO 1001
c--
c--     keep last mlblkl readings
c--     block numbers run from 0-199 for a 200 block file
c--     so to test for end must subtract one from file length
c--
510     ALBLK=0
        CALL BOURED(MKBLK,ALBLK,MKBLKW/2,MKBLKW,0,'RE','BO',LERR)
        IF (LERR.NE.0) GOTO 1106
        LBLK=ALBLK
        CALL BOURED(MLBLK,BOUDAT,MLBLKW/2,MLBLKW,LBLK,'WR','BO',LERR)
        IF (LERR.NE.0) GOTO 1107
        ALBLK=ALBLK+1
        IF (ALBLK.GT.(MLBLKL-1)) ALBLK=0
        CALL BOURED(MKBLK,ALBLK,MKBLKW/2,MKBLKW,0,'WR','BO',LERR)
        IF (LERR.NE.0) GOTO 1108
40      RETURN
c-----------------------------------------------------------------------
c--     error messages
c-----------------------------------------------------------------------
c--
1000    WRITE (LTERM,*)'DK1-ER MBBLK'
        GO TO 40
1001    WRITE (LTERM,*) 'DK1-EW MBBLK'
        GO TO 40
1005    WRITE (LTERM,*) 'DK1-EW MABLK'
        GOTO 80
1006    WRITE (LTERM,*) 'DK1-EW MMBLK'
        GOTO 81
1106    WRITE (LTERM,*) 'DK2-ER MKBLK'
        GO TO 40
1107    WRITE (LTERM,*) 'DK2-EW MLBLK'
        GO TO 40
1108    WRITE (LTERM,*) 'DK2-EW MKBLK'
        GO TO 40
        END
```

```
      SUBROUTINE BOUDK2
c +----------------------------------------------------------------------+
c | Program :     BOULD.REL                                              |
c | Module  :     BOUDK2.FOR                                             |
c | Function:     WRITES DATA TO DAILY ACCULATOR AND DATA FILES          |
c |                                                                      |
c +----------------------------------------------------------------------+
c--
c--   data common block
c--
      COMMON/BOULD/BOUDAT(60),BOUA(25),BOUB(25),BDATA(60),BOUT(60),
     *NARRAY,BCONAV(2),BTMPAV(2),BPRSAV(2),NPL,NPR,NBT1,NBT2,NBC1,
     *NBC2,NBP1,NBP2,LBCF(2),LBTF(2),LBPF(2),SETPTC(2),LOCB,
     *KFBR(2),KFBP,KSBR(2),KSBP,KTMPF,KCONF,KPRSF,KINTF,KWNDF,KFFTF,
     *KPUMP(2)
c--
c--   diagnostic block
c--
      LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
      INTEGER*4 I1AREA,I2AREA
      COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
     *LTERM,ITERM,ATIMES,BTIMES,IADPOK,IADPEK,IPHONE,ILDTR,IHDTR,
     *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
     *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
     *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
     *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
     *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
     *JBOVR,JBWAIT,JBSLEP
c--
c--   file information block
c--
      COMMON/FILE/BOUDEV,MABLK(4),MBBLK(4),MCBLK(4),MDBLK(4),
     *MEBLK(4),MFBLK(4),MGBLK(4),MHBLK(4),MIBLK(4),MJBLK(4),MKBLK(4),
     *MLBLK(4),MMBLK(4),MNBLK(4),MOBLK(4),MPBLK(4),MQBLK(4),MRBLK(4),
     *MSBLK(4),
     *MABLKL,MABLKW,MBBLKL,MBBLKW,MCBLKL,
     *MCBLKW,MDBLKL,MDBLKW,MEBLKL,MEBLKW,MFBLKL,MFBLKW,MGBLKL,
     *MGBLKW,MHBLKL,MHBLKW,MIBLKL,MIBLKW,MJBLKL,MJBLKW,MKBLKL,
     *MKBLKW,MLBLKL,MLBLKW,MMBLKL,MMBLKW,MNBLKL,MNBLKW,MOBLKL,
     *MOBLKW,MPBLKL,MPBLKW,MQBLKL,MQBLKW,MRBLKL,MRBLKW,MSBLKL,MSBLKW
c--
c-----------------------------------------------------------------------
c--
c--   diagnostic section
      CALL IPOKEB (JBDS1,JBDK1+1)
      CALL SCOPY (' BOUDK2 ',LEVEL2)
D     CALL PUTSTR(LTERM,' LOCATION: ','0')
D     CALL PUTSTR(LTERM,LEVEL1,' ')
D     CALL PUTSTR(LTERM,LEVEL2,' ')
D     CALL PUTSTR(LTERM,LEVEL3,' ')
c--
c--   initialize temporary array
```

```
c--     check to see if any file needs operated on
c--
        IF (SYSDAT.EQ.BOUDAT(59)) RETURN
c--
800     BOUDAT(59)=SYSDAT
c--
c--     clear out array   (be sure)
c--
        DO 44,I=1,NARRAY
44      BDATA(I)=0.
c--
c--
c--
        DO 810,I=1,25   !average only first 31 see boudat for assignments
        IF (BOUT(I).EQ.0.0) GOTO 810    !prevent zero divide
        BOUT(I)=BOUT(I)/BOUT(NARRAY)
810     BDATA(I)=BOUT(I)
c--
c--     keep readings
c--
        DO 811,I=30,44
811     BDATA(I)=BOUT(I)
c--
c--     ave of average readings
c--
        DO 812,I=45,50
        IF (BOUT(I).EQ.0.0) GOTO 812    !prevent zero divide
        BOUT(I)=BOUT(I)/BOUT(NARRAY)
812     BDATA(I)=BOUT(I)
c--
c--     keep readings for rest
c--
        DO 813,I=51,NARRAY
813     BDATA(I)=BOUT(I)
c--
c--     ave flow rate
c--
        IF (BDATA(51).EQ.0) GOTO 820
        BDATA(28)=BDATA(53)/BDATA(51)
820     IF (BDATA(52).EQ.0) GOTO 821
        BDATA(29)=BDATA(54)/BDATA(52)
c--
c-------------------------------------------
c--     get block number ,write to daily average file,update block number
c--     block number has to be read as a real*4 as this is the type of
c--     array boured puts the data in    [ ALBLK ]
c-------------------------------------------
c--
821     CALL BOURED(MEBLK,ALBLK,MEBLKW/2,MEBLKW,0,'RE','BO',LERR)
        IF (LERR.NE.0) GOTO 1100
        LBLK=ALBLK
        CALL BOURED(MDBLK,BDATA,MDBLKW/2,MDBLKW,LBLK,'WR','BO',LERR)
        IF (LERR.NE.0) GOTO 1101
        ALBLK=ALBLK+1
        IF (ALBLK.GT.(MDBLKL-1)) ALBLK=0
        CALL BOURED(MEBLK,ALBLK,MEBLKW/2,MEBLKW,0,'WR','BO',LERR)
        IF (LERR.NE.0) GOTO 1102
c--
c----------------------------------------
c--     weekly average (previous 7 days)
c----------------------------------------
c--
500     ALBLK=0
        CALL BOURED(MGBLK,ALBLK,MGBLKW/2,MGBLKW,0,'RE','BO',LERR)
        IF (LERR.NE.0) GOTO 1103
        LBLK=ALBLK
        CALL BOURED(MFBLK,BDATA,MFBLKW/2,MFBLKW,LBLK,'WR','BO',LERR)
        IF (LERR.NE.0) GOTO 1104
        ALBLK=ALBLK+1
        IF (ALBLK.GT.(MFBLKL-1)) ALBLK=0
        CALL BOURED(MGBLK,ALBLK,MGBLKW/2,MGBLKW,0,'WR','BO',LERR)
        IF (LERR.NE.0) GOTO 1105
c--
c----------------------------------------
c--     monthly average (previous 30 days)
c----------------------------------------
c--
510     ALBLK=0
        CALL BOURED(MIBLK,ALBLK,MIBLKW/2,MIBLKW,0,'RE','BO',LERR)
        IF (LERR.NE.0) GOTO 1106
        LBLK=ALBLK
        CALL BOURED(MHBLK,BDATA,MHBLKW/2,MHBLKW,LBLK,'WR','BO',LERR)
        IF (LERR.NE.0) GOTO 1107
        ALBLK=ALBLK+1
        IF (ALBLK.GT.(MHBLKL-1)) ALBLK=0
        CALL BOURED(MIBLK,ALBLK,MIBLKW/2,MIBLKW,0,'WR','BO',LERR)
        IF (LERR.NE.0) GOTO 1108
c--
c--     initialize temporary array
c--
        DO 35,I=1,NARRAY
35      BDATA(I)=0.
c--
c--     clear out accumulator file
c--
        CALL BOURED(MBBLK,BDATA,MBBLKW/2,MBBLKW,0,'WR','BO',LERR)
        IF (LERR.NE.0) GOTO 1110
41      RETURN
c--
c--------------------------------
c--     error statements and branching
c--------------------------------
c--
1100    WRITE (LTERM,*) 'DK2-ER MEBLK'
```

```
       GO TO 500
1101   WRITE (LTERM,*) 'DK2-EW MDBLK'
       GO TO 500
1102   WRITE (LTERM,*) 'DK2-EW MEBLK'
       GO TO 500
1103   WRITE (LTERM,*) 'DK2-ER MGBLK'
       GO TO 510
1104   WRITE (LTERM,*) 'DK2-EW MFBLK'
       GO TO 510
1105   WRITE (LTERM,*) 'DK2-EW MGBLK'
       GO TO 510
1106   WRITE (LTERM,*) 'DK2-ER MIBLK'
       GO TO 41
1107   WRITE (LTERM,*) 'DK2-EW MHBLK'
       GO TO 41
1108   WRITE (LTERM,*) 'DK2-EW MIBLK'
       GO TO 41
1110   WRITE (LTERM,*) 'DK2-EW MBBLK'
       GO TO 41
       END
```

```
      SUBROUTINE BOUER1
c +-----------------------------------------------------------------------+
c | Program :    BOULD.REL                                                |
c | Module  :    BOUER1.FOR                                               |
c | Function:    TEST DATA MODULES ERROR FLAGS TO DETERMINE               |
c |              SEVERITY OF DETECTED ERRORS                              |
c |                                                                       |
c +-----------------------------------------------------------------------+
c--
c--   data common block
c--
      COMMON/BOULD/BOUDAT(60),BOUA(25),BOUB(25),BDATA(60),BOUT(60),
      *NARRAY,BCONAV(2),BTMPAV(2),BPRSAV(2),NPL,NPR,NBT1,NBT2,NBC1,
      *NBC2,NBP1,NBP2,LBCF(2),LBTF(2),LBPF(2),SETPTC(2),LOCB,
      *KFBR(2),KFBP,KSBR(2),KSBP,KTMPF,KCONF,KPRSF,KINTF,KWNDF,KFFTF,
      *KPUMP(2)
c--
c--   channel switching common block
c--
      COMMON/CHAN/IBCONB,IBCOO,IBFCI,IBVFI,IBVFO,IBTCM,IBREFR,
      *IBWS,IBMPR,IBPPR,IBPNIC,MBCON(7),NBCON,MBTMP(10),
      *NBTMP,MBPMP(2),NBPMP,MBPRS(7),NBPRS,MBFFT(2),NBFFT,
      *MBCONR(3),NBCONR
c--
c--   control character common block
c--
      COMMON /CONTRL/LA,LB,LC,LR,LDLE,LQU,LLAS,LTACT,LLACT,LTAC,
      *LLAC,LI,LF,LCR,LLF,ITRH1,ITRM1,ITRS1,ITRT1,ITRT2,ITRT3
c--
c--   diagnostic block
c--
      LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
      INTEGER*4 I1AREA,I2AREA
      COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
      *LTERM,ITERM,ATIMES,BTIMES,IADPOK,IADPEK,IPHONE,ILDTR,IHDTR,
      *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
      *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
      *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
      *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
      *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
      *JBOVR,JBWAIT,JBSLEP
c--
c--   file information block
c--
      COMMON/FILE/BOUDEV,MABLK(4),MBBLK(4),MCBLK(4),MDBLK(4),
      *MEBLK(4),MFBLK(4),MGBLK(4),MHBLK(4),MIBLK(4),MJBLK(4),MKBLK(4),
      *MLBLK(4),MMBLK(4),MNBLK(4),MOBLK(4),MPBLK(4),MQBLK(4),MRBLK(4),
      *MSBLK(4),
      *MABLKL,MABLKW,MBBLKL,MBBLKW,MCBLKL,
      *MCBLKW,MDBLKL,MDBLKW,MEBLKL,MEBLKW,MFBLKL,MFBLKW,MGBLKL,
      *MGBLKW,MHBLKL,MHBLKW,MIBLKL,MIBLKW,MJBLKL,MJBLKW,MKBLKL,
      *MKBLKW,MLBLKL,MLBLKW,MMBLKL,MMBLKW,MNBLKL,MNBLKW,MOBLKL,
      *MOBLKW,MPBLKL,MPBLKW,MQBLKL,MQBLKW,MRBLKL,MRBLKW,MSBLKL,MSBLKW
c--
c--   bg diagnostic traceback
c--
      CALL IPOKEB (JBDS1,JBERR)
c--
c--   timing failure traceback
c--
      CALL SCOPY (' BOUERR',LEVEL2)
      CALL SCOPY ('       ',LEVEL3)
c--
c--   temporary debugging diagnostics
c--
D     CALL PUTSTR(LTERM,' LOCATION: ','0')
D     CALL PUTSTR(LTERM,LEVEL1,' ')
D     CALL PUTSTR(LTERM,LEVEL2,' ')
D     CALL PUTSTR(LTERM,LEVEL3,' ')
c-----------------------------------------------------------------------
c--
c--   test error flags
c--   if not error flags set return
c-----------------------------------------------------------------------
      IF (KCONF.EQ.0 .AND. KTMPF.EQ.0 .AND. KPRSF.EQ.0)
      * RETURN
c-----------------------------------------------------------------------
c--
c--   check all values to determine if scanner or measuring system faile
c--
c--   LVAR - total number of sensors (cond, temp, comm. cond.)
c--   IVAR - counter to find out how mant temp sensors are bad
c--          temp sensors are bad if all read 20. degrees C
c--
c--   Sensors for cond and comm. cond. are checked to see if they are
c--   0. This indicates that some piece of equipment is faulty
c--   possibly the measurement system
c-----------------------------------------------------------------------
c--
c--   nbtmp=number of temperatures
c--   1= oscillator
c--   nbprs=number of comm cond probes
c--
40    LVAR=(NBTMP+1+NBPRS)
      IVAR=0  !INIT TO 0
c--
c--   check temp sensors
c--   check comm.cond and cond probes and oscillator
c--
      DO 43 I=1,LVAR
43    IF (BOUDAT(I).EQ.0.) IVAR=IVAR+1
c--
c--   how many sensors were bad
c--   all ??  whole system down
```

```
c--     part??  check that part
c--
        IF (IVAR.NE.LVAR) RETURN
        WRITE (LTERM,42)
        RETURN
c--
c----------------------------------------------------------------------
c--
c--     section to test oscillator
c--
c----------------------------------------------------------------------
        IF (KCONF.EQ.0.) GOTO 20
60      IF (KCONF.GE.1) WRITE(LTERM,61)
        IF (KCONF.GT.1) WRITE (LTERM,66)          !IF 2
c----------------------------------------------------------------------
c--     temp module failure ?????
c--     possible scanner failure
c--     if temp mod failure make temp eq 20
c----------------------------------------------------------------------
20      IF (KTMPF.EQ.0) GOTO 30
        IF (KTMPF.GE.1) WRITE (LTERM,21)
        IF (KTMPF.GE.1) WRITE (LTERM,23)
        IF (KTMPF.GT.1) WRITE (LTERM,22)
c----------------------------------------------------------------------
c--
c--     section to test where conductance problem is in bouprs
c--
c----------------------------------------------------------------------
30      IF (KPRSF.EQ.0) GO TO 50
        IF (KPRSF.GE.1) WRITE(LTERM,31)
        IF (KPRSF.GT.1) WRITE (LTERM,36)
c----------------------------------------------------------------------
c--
c--     was temp only problem?
c--     check in second part of subroutine
c--
c----------------------------------------------------------------------
c--
50      RETURN
c----------------------------------------------------------------------
c--
c--     format statements
c--
c----------------------------------------------------------------------
42      FORMAT (' SCANNER OR MEASURING SYSTEM FAILURE')
61      FORMAT (' OSCILLATOR PROBLEM ')
66      FORMAT (' BAD OSCILLATOR')
23      FORMAT (' REFERENCE TEMPERATURE OUT OF RANGE')
21      FORMAT (' TEMPERATURE MODULE CHECK')
22      FORMAT (' TEMPERATURE MODULE FAILURE')
31      FORMAT (' CONDUCTANCE PROBLEM WITH COMMERCIAL PROBES')
36      FORMAT (' BAD OSCILLATOR')
```

END

```
      SUBROUTINE BOUER2
c +-------------------------------------------------------------------+
c | Program :    BOULD.REL                                            |
c | Module  :    BOUER2.FOR                                           |
c | Function:    SHUT DOWN SYSTEM IF SEVER ERROR IN DATA MODULES      |
c |                                                                   |
c +-------------------------------------------------------------------+
c--
c--   data common block
c--
      COMMON/BOULD/BOUDAT(60),BOUA(25),BOUB(25),BDATA(60),BOUT(60),
      *NARRAY,BCONAV(2),BTMPAV(2),BPRSAV(2),NPL,NPR,NBT1,NBT2,NBC1,
      *NBC2,NBP1,NBP2,LBCF(2),LBTF(2),LBPF(2),SETPTC(2),LOCB,
      *KFBR(2),KFBP,KSBR(2),KSBP,KTMPF,KCONF,KPRSF,KINTF,KWNDF,KFFTF,
      *KPUMP(2)
c--
c--   channel switching common block
c--
      COMMON/CHAN/IBCONB,IBCOO,IBFCI,IBVFI,IBVFO,IBTCM,IBREFR,
      *IBWS,IBMPR,IBPPR,IBPNIC,MBCON(7),NBCON,MBTMP(10),
      *NBTMP,MBPMP(2),NBPMP,MBPRS(7),NBPRS,MBFFT(2),NBFFT,
      *MBCONR(3),NBCONR
c--
c--   control character common block
c--
      COMMON /CONTRL/LA,LB,LC,LR,LDLE,LQU,LLAS,LTACT,LLACT,LTAC,
      *LLAC,LI,LF,LCR,LLF,ITRH1,ITRM1,ITRS1,ITRT1,ITRT2,ITRT3
c--
c--   diagnostic block
c--
      LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
      INTEGER*4 I1AREA,I2AREA
      COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
      *LTERM,ITERM,ATIMES,BTIMES,IADPOK,IADPEK,IPHONE,ILDTR,IHDTR,
      *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
      *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
      *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
      *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
      *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
      *JBOVR,JBWAIT,JBSLEP
c--
c--   file information block
c--
      COMMON/FILE/BOUDEV,MABLK(4),MBBLK(4),MCBLK(4),MDBLK(4),
      *MEBLK(4),MFBLK(4),MGBLK(4),MHBLK(4),MIBLK(4),MJBLK(4),MKBLK(4),
      *MLBLK(4),MMBLK(4),MNBLK(4),MOBLK(4),MPBLK(4),MQBLK(4),MRBLK(4),
      *MSBLK(4),
      *MABLKL,MABLKW,MBBLKL,MBBLKW,MCBLKL,
      *MCBLKW,MDBLKL,MDBLKW,MEBLKL,MEBLKW,MFBLKL,MFBLKW,MGBLKL,
      *MGBLKW,MHBLKL,MHBLKW,MIBLKL,MIBLKW,MJBLKL,MJBLKW,MKBLKL,
      *MKBLKW,MLBLKL,MLBLKW,MMBLKL,MMBLKW,MNBLKL,MNBLKW,MOBLKL,
      *MOBLKW,MPBLKL,MPBLKW,MQBLKL,MQBLKW,MRBLKL,MRBLKW,MSBLKL,MSBLKW
```

```
c--
c--
c--
        DATA IBELL /"007/
        KCNT=0
        KCP=0
c--
c--
c--
        CALL IPOKEB (JBDS1,JBERR+1)
c--
c--
c--
        CALL SCOPY (' BOUER2',LEVEL2)
        CALL SCOPY ('       ',LEVEL3)
c--
c--
c--
D       CALL PUTSTR(LTERM,' LOCATION: ','0')
D       CALL PUTSTR(LTERM,LEVEL1,' ')
D       CALL PUTSTR(LTERM,LEVEL2,' ')
D       CALL PUTSTR(LTERM,LEVEL3,' ')
c---------------------------------------------------
c--
c--     was temp only problem?
c--
c-----------------------------------------------------
50      IF (KTMPF.LE.1 .AND.KCONF.EQ.0 .AND.
        * KPRS.EQ.0) RETURN
c-----------------------------------------------
c--
c--     shut down system
c--
c-------------------------------------------
100     WRITE (LTERM,101)
101     FORMAT (///,' System shuting down to to failure of
        *equipment',/,' see BOULD.DAT for reason',///)
c------------------------------------------------------------
c--
c--     check to see if pumps on
c--     if on turn off
c--
c---------------------------------------------------------------------------
c--
c--     if override byte set do not shut down
c--
c---------------------------------------------------------------------------
c--     override turned on for debugging remove for normal execution
        IF (IPEEKB(JBDCH2).NE.JBOVR) GOTO 120
        WRITE (LTERM,*) 'Override byte set'
        RETURN
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c--
c--     see if any pumps need to be turned off
c--
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
120     IF (KINTF.GT.0) GOTO 9
c------------------------------------------------------------------------
c--
c--     no pumps need be off
c--     if 0 pump is off
c--     if 1 pump is on
c--
c------------------------------------------------------------------------
        RETURN
C???????????????????????????????????????????????????????????????????????
C???????????????????????????????????????????????????????????????????????
C???????????????????????????????????????????????????????????????????????
        IF (KFBP.EQ.0) GOTO 108
c------------------------------------------------------------------------
c--
c--     some pumps need be off
c--
c------------------------------------------------------------------------
150     CALL RLLTI(MFLAG)
        IF (MFLAG.EQ.1) GOTO 150
c--
        KPC=0   !init pump locator ie which pump to turn on/off
c--
c--     turn off main pump and plot pump
c--
        CALL BOUCHN(IBMPR,MFLAG)
        IF (MFLAG.EQ.1) GOTO 150
c--
c--     turn off plot pump
c--
        CALL BOUCHN(IBPPR,MFLAG)
        IF (MFLAG.EQ.1) GOTO 150
c--
c--     calculate which plot relay is to acted upon
c--
160     KPC=KPC+1
c--
c--     does this relay need turned off
c--     no  --) go check if that is all the relays
c--     yes --) yes turn off relay
c--
        IF (KFBR(KPC).EQ.1) GOTO 170
c--
c--     turn off that relay
c--
        CALL BOUCHN(KPUMP(L),MFLAG)
        IF (MFLAG.EQ.1) GOTO 150
c--
```

```
c--     is this all the relays?
c--     no --> do it again
c--     yes --> close the channels
c--
170     IF (KPC.LT.KCNT) GOTO 160
c--
c--     close the channels now
c--
        CALL RET(MFLAG)
        IF (MFLAG.EQ.1) GOTO 150
c--
c--     reset current state of pumps for future reference
c--
        BOUDAT(30)=0
        BOUDAT(31)=0
        BOUDAT(26)=0
        BOUDAT(27)=0
c---------------------------------------------------------
c--
c--     write to disk file information on pumps
c--
c---------------------------------------------------
        CALL BOURED (MABLK,BOUDAT,MABLKW/2,MABLKW,0,'WR','BO',IERR)
        IF (IERR.NE.0) GOTO 900
        GOTO 108
c--
c--     error on write to main disk file
c--
900     WRITE (LTERM,*) 'Error on main file write in BOUER2'
        WRITE (LTERM,*) 'Backup file was used'
        CALL BOURED (MMBLK,BOUDAT,MMBLKW/2,MMBLKW,0,'WR','BO',IERR)
        IF (IERR.EQ.0) GOTO 108
c--
c--     error on write for backup
c--     tell background program
c--
        WRITE (LTERM,*) 'Error on backup read. '
        IPOK=(IPEEKB(JBDERR))+JBEADD
        CALL IPOKEB(JBDERR,IPOK)
        GOTO 108
c--
c--     error flag set so no action taken
c--
9       WRITE (LTERM,*) 'No action in pump control due to
        *kintf flag set'
c--
c--
c-------------------------------------
c--
108     WRITE (LTERM,110) IBELL
110     FORMAT('$',A1,'System shutdown ! Error codes are:')
        WRITE (LTERM,*) 'ktmpf',KTMPF,' kconf',KCONF,' kprsf',KPRSF,
        *' kintf',KINTF
c--
c--     close output device
c--
        CALL CLOSE(LTERM)
        CALL ICMKT(0,I1AREA)     !CANCELS ALL TIMERS
c--
c--     send message to restrt program that shut down has occurred
c--
        CALL IPOKE(IPHONE,ILDTR)         !HANG UP PHONE
        CALL IPOKEB (JBDERR,JBESHT)      !SET ERROR BYTE
        CALL EXIT
        RETURN
        END
```

```
       SUBROUTINE FAIL(ID)
c +----------------------------------------------------------------------+
c | Program :    BOULD.REL                                               |
c | Module  :    BOUFAL.FOR                                              |
c | Function:    ROUTINE FOR SYSTEMATIC SHUTDOWN OF COMMUNICATIONS LINES |
c |              AND SYSYTEM PRIOR TO A RESTART. A TRACEBACK TO THE PROGR|
c |              LOCATION AT FAILURE TIME IS GIVEN                       |
c |                                                                      |
c +----------------------------------------------------------------------+
c--
c--    diagnostic block
c--
       LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
       INTEGER*4 I1AREA,I2AREA
       COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
       *LTERM,ITERM,ATIMES,BTIMES,IADPOK,IADPEK,IPHONE,ILDTR,IHDTR,
       *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
       *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
       *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
       *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
       *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
       *JBOVR,JBWAIT,JBSLEP
c--
c--    cancel any timers set
c--
       CALL ICMKT (0,I1AREA)   !CANCELL MAIN TIMER
       CALL ICMKT (0,I2AREA)   !CANCEL TIMER FROM DIAL
c--
c--    cancel any ctrl o that was set
c--
       CALL RCTRLO     !RESETS CTRL O
c--
c--    ring bell to get attention
c--
       DO 8 JK=1,5
8      CALL PRINT ("007,"200)
c--
c--    hang up phone
c--
       CALL IPOKE (IPHONE,IDTRL)
       ICH= ISLEEP (0,0,2,0)
c--
c--    give traceback history
c--
       WRITE(LTERM,9)
9      FORMAT (/,' **************************************************',/)
       WRITE(LTERM,10)
10     FORMAT (' TIMING FAILURE ')
       IF (ID.EQ.1) WRITE(LTERM,230)
       IF (ID.EQ.2) WRITE(LTERM,231)
       IF (ID.EQ.3) WRITE(LTERM,232)
230    FORMAT (' TIMER FAILURE IN MAIN PROGRAM')
```

```
231     FORMAT (' HANDSHAKE FAILURE IN BOURTL')
232     FORMAT (' TIMER FAILURE IN DIAL SECTION OF PROGRAM')
        WRITE(LTERM,52)
52      FORMAT (//,' THE HIERARCHY OF SUBROUTINES AT '/
        1' THE TIME OF FAILURE ARE: ')
        CALL PUTSTR (LTERM,LEVEL1)
        CALL PUTSTR (LTERM,LEVEL2)
        CALL PUTSTR (LTERM,LEVEL3)
        WRITE(LTERM,16)
16      FORMAT (//,' THE PROGRAM IS NOW BEING TERMINATED!!!!')
        WRITE(LTERM,500) SYSTM1,SYSDAT
500     FORMAT (X,'TIME :',F5.0,10X,'DATE  ',F7.0)
        WRITE(LTERM,33) BTIMES
33      FORMAT (/,' TIMES THRU BEFORE PROBLEM IS: ',F10.0)
        WRITE(LTERM,9)
        ICH= ISLEEP (0,0,2,0)
c--
c--     set restart byte so bgprogram will start fg program
c--
        CALL IPOKEB (JBDS1,JBFAL)       !SETS BIT SO PROGRAM RESTART
c--
c--     close output device and exit
c--
        CALL CLOSE (LTERM)
        CALL EXIT
        RETURN
        END
```

```
      SUBROUTINE BOUFFT
c +-----------------------------------------------------------------------+
c | Program :    BOULD.REL                                                |
c | Module  :    BOUFFT.FOR                                               |
c | Function:    READS FLOW FREQUENCY TRANSDUCER                          |
c |                                                                       |
c +-----------------------------------------------------------------------+
c--
c--     data common block
c--
      COMMON/BOULD/BOUDAT(60),BOUA(25),BOUB(25),BDATA(60),BOUT(60),
     *NARRAY,BCONAV(2),BTMPAV(2),BPRSAV(2),NPL,NPR,NBT1,NBT2,NBC1,
     *NBC2,NBP1,NBP2,LBCF(2),LBTF(2),LBPF(2),SETPTC(2),LOCB,
     *KFBR(2),KFBP,KSBR(2),KSBP,KTMPF,KCONF,KPRSF,KINTF,KWNDF,KFFTF,
     *KPUMP(2)
c--
c--     channel switching common block
c--
      COMMON/CHAN/IBCONB,IBCOO, IBFCI, IBVFI, IBVFO, IBTCM, IBREFR,
     *IBWS, IBMPR, IBPPR, IBPNIC,MBCON(7),NBCON, MBTMP(10),
     *NBTMP, MBPMP(2),NBPMP,MBPRS(7),NBPRS, MBFFT(2), NBFFT,
     *MBCONR(3),NBCONR
c--
c--     data checking common block
c--
      COMMON/CONDCL/WINDMX, TEMPMN, BCONL, BCONH, BCONRF(3), BRI(3),
     *BVOLTS, BAI, BTMPRF, BCONIN, BAVETP(2), BTMPHZ, RANL, RANH, BFTDAC(4)
c--
c--     control character common block
c--
      COMMON /CONTRL/LA, LB, LC, LR, LDLE, LQU, LLAS, LTACT, LLACT, LTAC,
     *LLAC, LI, LF, LCR, LLF, ITRH1, ITRM1, ITRS1, ITRT1, ITRT2, ITRT3
c--
c--     diagnostic block
c--
      LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
      INTEGER*4 I1AREA,I2AREA
      COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
     *LTERM, ITERM, ATIMES,BTIMES, IADPOK, IADPEK, IPHONE, ILDTR, IHDTR,
     *I1AREA(4), I2AREA(4), SYSTM1, SYSDAT, JBDIO, JBDPGM, JBDS1, JBDS2,
     *JBDCH1, JBDCH2, JBDCTL, JBDERR, JBDAL, JBTMP, JBWND, JBCON, JBPRS, JBFFT,
     *JBERR, JBMAH, JBPMP, JBSTA, JBTIM, JBDK1, JBSLP, JBPKC, JBINT,
     *JBRTL, JBCHN, JBRLL, JBWRT, JBRET, JBRED, JBBIT, JBFAL, JBDISK, JBTT,
     *JBECON, JBEINT, JBETMP, JBEFFT, JBEDAL, JBEHND, JBEADD, JBESHT, JBRUN,
     *JBOVR, JBWAIT, JBSLEP
c-------------------------------------------------------------------------
c--     diagnostic information
c--
c--     bg diagnostics
c--
      CALL IPOKEB(JBDS1, JBFFT)
c--
```

```
c--     timing failure traceback diagnostics
c--
        CALL SCOPY (' BOUFFT',LEVEL2)
        CALL SCOPY ('       ',LEVEL3)
c--
c--     temporary diagnostics
c--
D       CALL PUTSTR(LTERM,' LOCATION: ','0')
D       CALL PUTSTR(LTERM,LEVEL1,' ')
D       CALL PUTSTR(LTERM,LEVEL2,' ')
D       CALL PUTSTR(LTERM,LEVEL3,' ')
c--
c-----------------------------------------------------------------------
c--
c--
c--
        DO 30,L=1,NBFFT
        ITRY=0  !init zero check
        LOCB=27+L        !array location for boudat loc (28,29)
c--
c--     start control sequence
c--
20      CALL RLLTI(MFLAG)        !set conditions so channels can be closed
        IF (MFLAG.EQ.1) GO TO 20
c--
c--     close frequency counter input
c--
        CALL BOUCHN (IBFCI,MFLAG)
        IF (MFLAG.EQ.1) GO TO 20
c--
c--     close frequency transducer for plot
c--
        CALL BOUCHN (MBFFT(L),MFLAG)
        IF (MFLAG.EQ.1) GO TO 20
c--
c--     finish control sequence
c--
        CALL RET(MFLAG) !tells scanner all commands have been sent
        IF (MFLAG.EQ.1) GO TO 20
c--
        CALL RTLLI(MFLAG)        !sets conditions so data can be received
        IF (MFLAG.EQ.1) GO TO 20
c--
        CALL WRITE(BOUDAT(LOCB),MFLAG)  !converts data to real*4 number
        IF (MFLAG.EQ.1) GO TO 20
c--
c--     check for zero condition
c--     a zero may not be a zero but a bad reading ckeck and see
c--
        ITRY=ITRY+1
c--
c--     is the flow 0? it will be if all pumps are off
```

```
c--     use this method instead of pumps flags because
c--     pumps could have been turned off when supposed to be on
c--     this method will pick it up.
c--
        IF (BOUDAT(LOCB).EQ.0. .AND. (BFTDAC(3) .EQ. 0)
        * .AND. (BFTDAC(4) .EQ. 0)) GOTO 30
        IF (ITRY.LE.5 .AND.BOUDAT(LOCB) .EQ. 0.) GOTO 20
c--
c--     check for zero if so branch,if not convert to freq.
c--
c--
c--     print out results for debug
c--
D       WRITE(LTERM,9000) BOUDAT(LOCB),LOCB,LEVEL2
9000    FORMAT (X,5X,'READING',E12.5,5X,'LOC',I5,10A1)
c--
c--     convert from frequency to gallons per minute
c--
        BOUDAT(LOCB)=BOUDAT(LOCB)*100.
30      CONTINUE
        RETURN
        END

        SUBROUTINE BOUIN1
c +------------------------------------------------------------------+
c | Program :    BOULD.REL                                           |
c | Module  :    BOUIN1.FOR                                          |
c | Function:    CHECKS DEVICE ASSIGNMENT AND READS INITIAL DATA FILES |
c +------------------------------------------------------------------+
c--
c--     data common block
c--
        COMMON/BOULD/BOUDAT(60),BOUA(25),BOUB(25),BDATA(60),BOUT(60),
        *NARRAY,BCONAV(2),BTMPAV(2),BPRSAV(2),NPL,NPR,NBT1,NBT2,NBC1,
        *NBC2,NBP1,NBP2,LBCF(2),LBTF(2),LBPF(2),SETPTC(2),LOCB,
        *KFBR(2),KFBP,KSBR(2),KSBP,KTMPF,KCONF,KPRSF,KINTF,KWNDF,KFFTF,
        *KPUMP(2)
c--
c--     diagnostic block
c--
        LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
        INTEGER*4 I1AREA,I2AREA
        COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
        *LTERM,ITERM,ATIMES,BTIMES,IADPOK,IADPEK,IPHONE,ILDTR,IHDTR,
        *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
        *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
        *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
        *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
        *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
        *JBOVR,JBWAIT,JBSLEP
c--
c--     file information block
```

```
C--
      COMMON/FILE/BOUDEV,MABLK(4),MBBLK(4),MCBLK(4),MDBLK(4),
     *MEBLK(4),MFBLK(4),MGBLK(4),MHBLK(4),MIBLK(4),MJBLK(4),MKBLK(4),
     *MLBLK(4),MMBLK(4),MNBLK(4),MOBLK(4),MPBLK(4),MQBLK(4),MRBLK(4),
     *MSBLK(4),
     *MABLKL,MABLKW,MBBLKL,MBBLKW,MCBLKL,
     *MCBLKW,MDBLKL,MDBLKW,MEBLKL,MEBLKW,MFBLKL,MFBLKW,MGBLKL,
     *MGBLKW,MHBLKL,MHBLKW,MIBLKL,MIBLKW,MJBLKL,MJBLKW,MKBLKL,
     *MKBLKW,MLBLKL,MLBLKW,MMBLKL,MMBLKW,MNBLKL,MNBLKW,MOBLKL,
     *MOBLKW,MPBLKL,MPBLKW,MQBLKL,MQBLKW,MRBLKL,MRBLKW,MSBLKL,MSBLKW
C--
C--   main program sleep block
C--
      COMMON/WAIT/IBPOLH,IBPOLM,IBPOLS,IBPOLT,IBCC,IBCH,IBCM,
     *IBCS,IBCT
C-----------------------------------------------------------------------
C--
C--   dimension for handler checking and time demension array
C--
      DIMENSION ICBLK(4),ZWRITE(2)
C--
C--   data declaration
C--
      DATA ICBLK/4*0/
C--
C--   bg diagnostics
C--
      CALL IPOKEB (JBDS1,JBINT)
C--
C--   timing failure traceback
C--
      CALL SCOPY (' BOUIN1',LEVEL2)
      CALL SCOPY ('       ',LEVEL3)
C--
C--   temporary diagnostics for debugging
C--
D     CALL PUTSTR(LTERM,' LOCATION: ','0')
D     CALL PUTSTR(LTERM,LEVEL1,' ')
D     CALL PUTSTR(LTERM,LEVEL2,' ')
D     CALL PUTSTR(LTERM,LEVEL3,' ')
C--
C-----------------------------------------------------------------------
C--   make sure handler is loaded
C-----------------------------------------------------------------------
C--
      CALL IDSTAT (BOUDEV,ICBLK)      !make sure device handler is loadd
      IF (ICBLK(3).NE.0)GO TO 500
      ITEST=IFETCH(BOUDEV)
      WRITE (LTERM,*) 'HANDLER BEING FETCHED'
C--
C--   error on loading handler  stop program
C--
```

```
      IF(ITEST.EQ.0) GO TO 500
      IF(ITEST.EQ.1) WRITE (LTERM,501)
      IF(ITEST.EQ.2) WRITE (LTERM,502)
      IF (ITEST.EQ.3) WRITE (LTERM,503)
      WRITE (LTERM,*) 'ASSIGN "BOU" TO SAME DISK PROGRAM IS ON'
      WRITE (LTERM,*) 'BE SURE DATA FILES ARE ALSO ON THIS DISK'
      STOP 'ERROR LOADING DEVICE HANDLER'
c--
c-----------------------------------------------------------------------
c--   read disk file and get pump status information
c-----------------------------------------------------------------------
c--
500   CALL BOURED (MABLK,BOUT,MABLKW/2,MABLKW,0,'RE','BO',IERR)
      IF (IERR.NE.0) GOTO 900
c--
c--   assign current pump staus to pump flags
c--   seee BOUDAT.for for variable names
c--
9     KFBR(1)=BOUT(30)
      KFBR(2)=BOUT(31)
      KFBP=BOUT(26)
      DO 8,I=1,NARRAY
8     BOUDAT(I)=BOUT(I)
      RETURN
c--
c--   error on read for pump status
c--
900   WRITE (LTERM,*) 'ERROR ON MAIN FILE READ IN BOUIN1'
      WRITE (LTERM,*) 'BACKUP FILE WAS READ'
c--
c--   on error read backup file
c--
      CALL BOURED (MMBLK,BOUT,MMBLKW/2,MMBLKW,0,'RE','BO',IERR)
c--
c--   backup ok continue
c--
      IF (IERR.EQ.0) GOTO 9
c--
c--   error with backup also
c--
      WRITE (LTERM,*) 'ERROR ON BACKUP READ. '
      KINTF=1 !set error flag if error on backup skip pump section
      WRITE (LTERM,*) 'ERROR FLAG SET SO PUMPS CAN NOT
     *BE TURNED ON OR OFF!!!!'
c--
      IPOK=(IPEEKB(JBDERR))+JBEINT
      CALL IPOKEB(JBDERR,IPOK)
      RETURN
c--
c--   format statements
c--
501   FORMAT(' DEVICE NAME SPECIFIED DOES NOT EXIST')
```

```
502     FORMAT (' NOT ENOUGH MEMOREY TO LOAD HANDLER')
503     FORMAT (' NO HANDLER EXISTS ON THE SYSTEM DEVICE')
        END
```

```
      SUBROUTINE BOUIN2
c +--------------------------------------------------------------------+
c | Program :    BOULD.REL                                              |
c | Module  :    BOUIN2.FOR                                             |
c | Function:    DETERMINES CPU RUNNING ON AND DATE AND TIME OF NEXT RUN |
c |                                                                     |
c +--------------------------------------------------------------------+
c--
c--      data common block
c--
         COMMON/BOULD/BOUDAT(60),BOUA(25),BOUB(25),BDATA(60),BOUT(60),
         *NARRAY,BCONAV(2),BTMPAV(2),BPRSAV(2),NPL,NPR,NBT1,NBT2,NBC1,
         *NBC2,NBP1,NBP2,LBCF(2),LBTF(2),LBPF(2),SETPTC(2),LOCB,
         *KFBR(2),KFBP,KSBR(2),KSBP,KTMPF,KCONF,KPRSF,KINTF,KWNDF,KFFTF,
         *KPUMP(2)
c--
c--      diagnostic block
c--
         LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
         INTEGER*4 I1AREA,I2AREA
         COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
         *LTERM, ITERM, ATIMES,BTIMES, IADPOK, IADPEK, IPHONE, ILDTR, IHDTR,
         *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
         *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
         *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
         *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
         *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
         *JBOVR,JBWAIT,JBSLEP
c--
c--      file information block
c--
         COMMON/FILE/BOUDEV,MABLK(4),MBBLK(4),MCBLK(4),MDBLK(4),
         *MEBLK(4),MFBLK(4),MGBLK(4),MHBLK(4),MIBLK(4),MJBLK(4),MKBLK(4),
         *MLBLK(4),MMBLK(4),MNBLK(4),MOBLK(4),MPBLK(4),MQBLK(4),MRBLK(4),
         *MSBLK(4),
         *MABLKL,MABLKW,MBBLKL,MBBLKW,MCBLKL,
         *MCBLKW,MDBLKL,MDBLKW,MEBLKL,MEBLKW,MFBLKL,MFBLKW,MGBLKL,
         *MGBLKW,MHBLKL,MHBLKW,MIBLKL,MIBLKW,MJBLKL,MJBLKW,MKBLKL,
         *MKBLKW,MLBLKL,MLBLKW,MMBLKL,MMBLKW,MNBLKL,MNBLKW,MOBLKL,
         *MOBLKW,MPBLKL,MPBLKW,MQBLKL,MQBLKW,MRBLKL,MRBLKW,MSBLKL,MSBLKW
c--
c--      base address common block
c--
         COMMON/PROCES/IAD103,IAD123,ITM03,ITM23
c--
c--      main program sleep block
c--
         COMMON/WAIT/IBPOLH,IBPOLM,IBPOLS,IBPOLT,IBCC,IBCH,IBCM,
         *IBCS,IBCT
c--
c--
c--
```

```
c--     bits common
c--
        COMMON/BITS/IBIT(16),IWORD
c--
c--
c--     time array to store date and time
c--
        DIMENSION ZWRITE(2)
c--
c--     bg diagnostics traceback
c--
        CALL IPOKEB (JBDS1,JBINT+1)
c--
c--     timing failure traceback
c--
        CALL SCOPY (' BOUIN2',LEVEL2)
        CALL SCOPY ('       ',LEVEL3)
c--
c--     tempory diagnostics for debugging
c--
D       CALL PUTSTR(LTERM,' LOCATION: ','0')
D       CALL PUTSTR(LTERM,LEVEL1,' ')
D       CALL PUTSTR(LTERM,LEVEL2,' ')
D       CALL PUTSTR(LTERM,LEVEL3,' ')
c-----------------------------------------
c--
c--     determine if have to wait before starting program
c--
c--------------------------------------------
c--
c--     read disk file for time and date
c--
10      CALL BOURED (MJBLK,ZWRITE,MJBLKW/2,MJBLKW,0,'RE','BO',LERR)
        IF (LERR.NE.0) GOTO 20
c--
c--     initalize time flag and set time and date
c--
        IBCC=0  !initialize time flag
        TIME=ZWRITE(1)
        DATE=ZWRITE(2)
c--
c--     first time thru no date and time
c--
        IF (TIME.EQ.0..AND.DATE.EQ.0.) GOTO 30
c--
c--     calculte diff in dates
c--
c--     not same day go and run
c--
        IF (SYSDAT-DATE.NE.0) GOTO 30
c--
c--     compare with when you are supposed to run again
```

```
c--
c--     compare time how long has it been
c--
        IF (SYSTM1 .GT.TIME) GOTO 30
        TDIF=TIMDIF(TIME,SYSTM1)        !diff in times from disk  system
c--
c--     (0 no wait go and run
c--
c--
c--
c--     calculate how long you are supposed to wait
c--
        IBCC=1  !set new time flag
        IBCH=(TDIF)/100.          !hours
        IBCM=(TDIF)-(FLOAT(IBCH)*100.)  !minutes
c--
c--
c--     tell bg you are in a wait mode
c--
        CALL IPOKEB (JBDPGM,JBWAIT)
        GOTO 30
c--
c--     error on write
c--
20      WRITE (LTERM,*) 'ER IN2 MJBLK'
c------------------------------------------
c--
c--     determine configuration of system for address assignments
c--
c--------------------------------------
30      IWORD=ISPY("300)        !main configuration word offset 300
        CALL BOUBIT
c--
c--     check bit assinments to determine operating conditions
c--     remember bits assignments shifted
c--     ex bit 0 =ibit(1)
c--
        IF( IBIT(1).NE.1) GOTO 200      !check monitor
        IF (IBIT(6) .NE.0) WRITE (LTERM,*) 'Timing was developed
        *using a 60 cycle clock'
        IF (IBIT(10) .NE.1) WRITE (LTERM,*) 'USR is not permanent'
        IF (IBIT(12).EQ.1) GOTO 300
c--
c--     set conditions for 1123 processor
c--
        IPHONE=IAD123
        IADPEK=IAD123+2
        IADPOK=IAD123+6
        ITERM=ITM23
D       WRITE (LTERM,*) 'This is a 11/23 processor'
D       WRITE (LTERM,*) 'Proper address assignments are being made'
D       WRITE (LTERM,1) IPHONE,ITERM
```

```
        RETURN
c--
c--     set conditions for 1103 processor
c--
300     IPHONE=IAD103
        IADPEK=IAD103+2
        IADPOK=IAD103+6
        ITERM=ITM03
D       WRITE (LTERM,*) 'This is a 11/03 processor.'
D       WRITE (LTERM,*) 'Proper address assignments are being made'
D       WRITE (LTERM,1) IPHONE,ITERM
        RETURN
c--
c--     wrong monitor --> exit
c--
200     WRITE (LTERM,*) 'You are not running under the
        *foreground or extened memory monitor !'
        CALL EXIT
c--
c--     format statements
c--
D1      FORMAT (' Base address: ',O6,10X,'Remote terminal: ',I3,//)
        END
```

```
      SUBROUTINE BOUIN3
c +------------------------------------------------------------------------+
c | Program :    BOULD.REL                                                 |
c | Module  :    BOUIN3.FOR                                                |
c | Function:    ROUTINE TO READ CHANGEABLE VALUES FROM BSTATS.DAT         |
c |                                                                        |
c +------------------------------------------------------------------------+
c--
c--     data common block
c--
        COMMON/BOULD/BOUDAT(60),BOUA(25),BOUB(25),BDATA(60),BOUT(60),
        *NARRAY,BCONAV(2),BTMPAV(2),BPRSAV(2),NPL,NPR,NBT1,NBT2,NBC1,
        *NBC2,NBP1,NBP2,LBCF(2),LBTF(2),LBPF(2),SETPTC(2),LOCB,
        *KFBR(2),KFBP,KSBR(2),KSBP,KTMPF,KCONF,KPRSF,KINTF,KWNDF,KFFTF,
        *KPUMP(2)
c--
c--     channel switching common block
c--
        COMMON/CHAN/IBCONB,IBCOO, IBFCI, IBVFI, IBVFO, IBTCM, IBREFR,
        *IBWS, IBMPR, IBPPR, IBPNIC,MBCON(7),NBCON, MBTMP(10),
        *NBTMP, MBPMP(2),NBPMP,MBPRS(7),NBPRS,MBFFT(2),NBFFT,
        *MBCONR(3),NBCONR
c--
c--     data checking common block
c--
        COMMON/CONDCL/WINDMX,TEMPMN,BCONL,BCONH,BCONRF(3),BRI(3),
        *BVOLTS,BAI,BTMPRF,BCONIN,BAVETP(2),BTMPHZ,RANL,RANH,BFTDAC(4)
c--
c--     control character common block
c--
        COMMON /CONTRL/LA,LB,LC,LR,LDLE,LQU,LLAS,LTACT,LLACT,LTAC,
        *LLAC,LI,LF,LCR,LLF,ITRH1,ITRM1,ITRS1,ITRT1,ITRT2,ITRT3
c--
c--     diagnostic block
c--
        LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
        INTEGER*4 I1AREA,I2AREA
        COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
        *LTERM, ITERM, ATIMES,BTIMES, IADPOK, IADPEK, IPHONE, ILDTR, IHDTR,
        *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
        *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
        *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
        *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
        *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
        *JBOVR,JBWAIT,JBSLEP
c--
c--     file information block
c--
        COMMON/FILE/BOUDEV,MABLK(4),MBBLK(4),MCBLK(4),MDBLK(4),
        *MEBLK(4),MFBLK(4),MGBLK(4),MHBLK(4),MIBLK(4),MJBLK(4),MKBLK(4),
        *MLBLK(4),MMBLK(4),MNBLK(4),MOBLK(4),MPBLK(4),MQBLK(4),MRBLK(4),
        *MSBLK(4),
        *MABLKL, MABLKW, MBBLKL, MBBLKW, MCBLKL,
        *MCBLKW, MDBLKL, MDBLKW, MEBLKL, MEBLKW, MFBLKL, MFBLKW, MGBLKL,
        *MGBLKW, MHBLKL, MHBLKW, MIBLKL, MIBLKW, MJBLKL, MJBLKW, MKBLKL,
        *MKBLKW, MLBLKL, MLBLKW, MMBLKL, MMBLKW, MNBLKL, MNBLKW, MOBLKL,
        *MOBLKW, MPBLKL, MPBLKW, MQBLKL, MQBLKW, MRBLKL, MRBLKW, MSBLKL, MSBLKW
c--
c--     telephone number common block
c--
        LOGICAL*1 NUMBOU,NUMADA
        COMMON/NUMBER/NUMBOU(13),NUMADA(9)
c--
c--     base address common block
c--
        COMMON/PROCES/IAD103, IAD123, ITM03, ITM23
c--
c--     timer failure common block
c--
        COMMON/TIMER/ITMH, ITMM, ITMS, ITMT, ITDH, ITDM, ITDS, ITST
c--
c--     main program sleep block
c--
        COMMON/WAIT/IBPOLH, IBPOLM, IBPOLS, IBPOLT, IBCC, IBCH, IBCM,
        *IBCS, IBCT
c-----------------------------------------------------------------------
c--     dimensions for this routine
c--
        DIMENSION TEMP(60)
c--
c--     diagnostics
c--
c--     bg diagnostic byte
c--
        CALL IPOKEB (JBDS1,JBINT+2)
c--
c--     timing failure traceback
c--
        CALL SCOPY (' BOUIN3',LEVEL2)
        CALL SCOPY ('       ',LEVEL3)
c--
c--     temporary diagnostics for debugging
c--
D       CALL PUTSTR(LTERM,' LOCATION: ','0')
D       CALL PUTSTR(LTERM,LEVEL1,' ')
D       CALL PUTSTR(LTERM,LEVEL2,' ')
D       CALL PUTSTR(LTERM,LEVEL3,' ')
c--
c--
c-----------------------------------------------------------------------
c--
c--     data assignments in array TEMP
c-----------------------------------------------------------------------
c--
```

```
c--     setptc(1)       temp(1)         set point plot 1
c--     setptc(2)       temp(2)         set point plot 2
c--     windmx          temp(5)         max wind speed
c--     tempmn          temp(6)         minimum temperature
c--     bavetp(1)       temp(7)         temp set if error in temp module
c--     bavetp(2)       temp(8)         temp set if error in temp module
c--     itmh            temp(11)        main timer - hours
c--     itmm            temp(12)        main timer - minutes
c--     itdh            temp(13)        dialer timer - hours
c--     itdm            temp(14)        dialer timer - minutes
c--     ibpolh          temp(20)        wait or sleep time - hours
c--     ibpolm          temp(21)        wait or sleep time - minutes
c--
c--     other array locations are unassigned at this time
c--     see BOUDAT.FOR for more information about each variable
c--             or their default values
c--
c------------------------------------------------------------------------
c--
c--     read data file
c--
        CALL BOURED(MNBLK,TEMP,MNBLKW/2,MNBLKW,0,'RE','BO',LERR)
c--
c--     check for error
c--     if error ignore rest and return
c--
        IF (LERR.EQ.0) GOTO 100
        WRITE (LTERM,*) 'Error on reading setpoints. Default
        * values to be used'
        RETURN
c--
c--     assign data values
c--
100     SETPTC(1)=TEMP(1)
        SETPTC(2)=TEMP(2)
c--
c--     save setpoints in data file
c--
        BOUDAT(53)=SETPTC(1)
        BOUDAT(54)=SETPTC(2)
c--
c--
        WINDMX=TEMP(5)
        TEMPMN=TEMP(6)
        BAVETP(1)=TEMP(7)
        BAVETP(2)=TEMP(8)
        ITMH=TEMP(11)
        ITMM=TEMP(12)
        ITDH=TEMP(13)
        ITDM=TEMP(14)
        IBPOLH=TEMP(20)
        IBPOLM=TEMP(21)
```

```
c--
          RETURN
          END
```

```
      SUBROUTINE BOUMH1
c +-----------------------------------------------------------------+
c | Program :     BOULD.REL                                         |
c | Module  :     BOUMH1.FOR                                        |
c | Function:     CALCULATE AVERAGE CONDUCTIVITY AND DETERMINE IF ANY PROB|
c |               ARE OUT OF RANGE                                  |
c |                                                                 |
c +-----------------------------------------------------------------+
c +-----------------------------------------------------------------+
c |   NOTE:                                                         |
c |                                                                 |
c |     determination is done by assigning the values 1,2, or 4 to probes|
c |      1,2,3 for a given plot .These values are then accumulated  |
c |                                                                 |
c |         value         probes                                    |
c |         ~~~~~         ~~~~~~                                    |
c |           0            none                                     |
c |           1             1                                       |
c |           2             2                                       |
c |           3             1  2                                    |
c |           4             3                                       |
c |           5             1  3                                    |
c |           6             2  3                                    |
c |           7             1,  2,   3 (all)                        |
c |                                                                 |
c +-----------------------------------------------------------------+
c--
c--     data common block
c--
      COMMON/BOULD/BOUDAT(60),BOUA(25),BOUB(25),BDATA(60),BOUT(60),
     *NARRAY,BCONAV(2),BTMPAV(2),BPRSAV(2),NPL,NPR,NBT1,NBT2,NBC1,
     *NBC2,NBP1,NBP2,LBCF(2),LBTF(2),LBPF(2),SETPTC(2),LOCB,
     *KFBR(2),KFBP,KSBR(2),KSBP,KTMPF,KCONF,KPRSF,KINTF,KWNDF,KFFTF,
     *KPUMP(2)
c--
c--     data checking common block
c--
      COMMON/CONDCL/WINDMX,TEMPMN,BCONL,BCONH,BCONRF(3),BRI(3),
     *BVOLTS,BAI,BTMPRF,BCONIN,BAVETP(2),BTMPHZ,RANL,RANH,BFTDAC(4)
c--
c--     diagnostic block
c--
      LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
      INTEGER*4 I1AREA,I2AREA
      COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
     *LTERM,ITERM,ATIMES,BTIMES,IADPOK,IADPEK,IPHONE,ILDTR,IHDTR,
     *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
     *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
     *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
     *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
     *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
     *JBOVR,JBWAIT,JBSLEP
```

```
c--
c--     set bg diagnostics
c--
        CALL IPOKEB (JBDS1,JBMAH)
c--
c--     timing failure traceback
c--
        CALL SCOPY (' BOUMH1',LEVEL2)
        CALL SCOPY ('       ',LEVEL3)
c--
c--     tempory diagnostics for debugging
c--
D       CALL PUTSTR(LTERM,' LOCATION: ','0')
D       CALL PUTSTR(LTERM,LEVEL1,' ')
D       CALL PUTSTR(LTERM,LEVEL2,' ')
D       CALL PUTSTR(LTERM,LEVEL3,' ')
c------------------------------------------------------------------------
c--
c--      calculate average for conductivities
c------------------------------------------------------------------------
c--
c--     init flags for out of range
c--
        DO 9,JH=1,NPL
        LBCF(JH)=0
9       LBPF(JH)=0
c--
c--     sum numbers
c--
c--
c--     do for number of plots and number of probes
c--
        DO 14 J=1,NPL   !PLOTS
        NPRC=0
        NPRP=0
c--
c--     iz. is where in the data array to store number
c--     this is done this way so if more data and/or sensors are added
c--     the array locations would automatically be corrected
c--       -1 because you are added the value of the do loop to it
c--        this makes sure that all the values are in ther proper locatios
c--
        IF (J.EQ.1) IZC=NBC1-1
        IF (J.EQ.2) IZC=NBC2-1
        IF (J.EQ.1) IZP=NBP1-1
        IF (J.EQ.2) IZP=NBP2-1
c--
c--     do calculations for number of probes
c--
        DO 10,I=1,NPR   !PROBES
c--
c--     check to see if readings fall within the specified range
```

```
c--     this range is specified in boudat.for
c--     do this by number of plots and number off probes
c--
c--
c--     conductivity probes
c--
            IY=MOD (4,(4-I))
        IF (BOUDAT(I+IZC).GE.BOUA(I+IZC).AND.
        * BOUDAT(I+IZC).LE.BOUB(I+IZC)) GOTO 12
          LBCF(J)=LBCF(J)+IY
D       WRITE (LTERM,*) 'COND CHECK',BOUA(I+IZC),BOUDAT(I+IZC),BOUB(I+IZC)
        GOTO 15
12      BCONAV(J)=BCONAV(J)+BOUDAT(I+IZC)
        NPRC=NPRC+1
c--
c--     commercial conductivity probes
c--
15      IF (BOUDAT(I+IZP).GE.BOUA(I+IZP).AND.
        *  BOUDAT(I+IZP).LE.BOUB(I+IZP)) GOTO 22
        LBPF(J)=LBPF(J)+IY
D       WRITE (LTERM,*) 'COMM COND CHECK',BOUA(I+IZC),BOUDAT(I+IZC),BOUB()
        GOTO 10
22      BPRSAV(J)=BPRSAV(J)+BOUDAT(I+IZP)
        NPRP=NPRP+1
10      CONTINUE
c--
c--     now that we have the sums lets do the averages
c--
D       WRITE (LTERM,*) 'NPRP',NPRP,'NPRC',NPRC
        IF (NPRC.EQ.0) GOTO 300
        BCONAV(J)=BCONAV(J)/NPRC
300     IF (NPRP.EQ.0) GOTO 14
        BPRSAV(J)=BPRSAV(J)/NPRP
14      CONTINUE
c--
c--     continue calculations and testing in second part
c--
c--
        DO 44,I=1,NPL
44      IF (LBCF(I).EQ.0) GOTO 43
        WRITE (LTERM,60) I,LBCF(I)
43      IF (LBPF(I).EQ.0) GOTO 45
        WRITE (LTERM,61) I,LBPF(I)
45      CONTINUE
        RETURN
c-----------------------------------------------------------------------
c--
c--     format statements
c--
c-----------------------------------------------------------------------
60      FORMAT (' PLOT ',I1,X,'CONDUCTIVITY PROBE ERROR CODE: ',I5)
61      FORMAT (' PLOT ',I1,X,'COMMERCIAL CONDUCTIVITY
* PROBE ERROR CODE: ',I5)
END
```

```
      SUBROUTINE BOUMH2
c +-------------------------------------------------------------------+
c | Program :    BOULD.REL                                            |
c | Module  :    BOUMH2.FOR                                           |
c | Function:    CHECKS WIND SPEED AND TEMPERATURE TO DETERMINE IF IN |
c |              APPLICATION RANGE. SETS FLAGS TO TURN ON PUMPS IF NEEDED|
c |                                                                   |
c +-------------------------------------------------------------------+
c--
c--   data common block
c--
      COMMON/BOULD/BOUDAT(60),BOUA(25),BOUB(25),BDATA(60),BOUT(60),
     *NARRAY,BCONAV(2),BTMPAV(2),BPRSAV(2),NPL,NPR,NBT1,NBT2,NBC1,
     *NBC2,NBP1,NBP2,LBCF(2),LBTF(2),LBPF(2),SETPTC(2),LOCB,
     *KFBR(2),KFBP,KSBR(2),KSBP,KTMPF,KCONF,KPRSF,KINTF,KWNDF,KFFTF,
     *KPUMP(2)
c--
c--
c--   channel switching common block
c--
      COMMON/CHAN/IBCONB,IBCOO,IBFCI,IBVFI,IBVFO,IBTCM,IBREFR,
     *IBWS,IBMPR,IBPPR,IBPNIC,MBCON(7),NBCON,MBTMP(10),
     *NBTMP,MBPMP(2),NBPMP,MBPRS(7),NBPRS,MBFFT(2),NBFFT,
     *MBCONR(3),NBCONR
c--
c--   data checking common block
c--
      COMMON/CONDCL/WINDMX,TEMPMN,BCONL,BCONH,BCONRF(3),BRI(3),
     *BVOLTS,BAI,BTMPRF,BCONIN,BAVETP(2),BTMPHZ,RANL,RANH,BFTDAC(4)
c--
c--   diagnostic block
c--
      LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
      INTEGER*4 I1AREA,I2AREA
      COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
     *LTERM,ITERM,ATIMES,BTIMES,IADPOK,IADPEK,IPHONE,ILDTR,IHDTR,
     *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
     *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
     *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
     *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
     *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
     *JBOVR,JBWAIT,JBSLEP
c-------------------------------------------------------------------
c--
c--   dimensioning for routine
c--
c--   SET  - array to hold Boudat(?) for setpoint check
c--   ITAB - array which holds probe status information for range check
c--          if LCBF matches a number in this array probe is bad
c--          each probe may have different numbers in this array
c--          that is probe 1 will differ from probe 2 and probe 3
c------------------------------------------------------------------
```

```
        DIMENSION SET(2),ITAB(2,4)
c--
c--     array is filled out in order itab(1,1), itab(2,1), itab(1,2), etc
c--
        DATA ITAB/1,1,3,3,5,5,7,7/
c--
c--     bg diagnostic byte
c--
        CALL IPOKEB (JBDS1,JBMAH+1)
c--
c--     timing failure traceback
c--
        CALL SCOPY (' BOUMH2',LEVEL2)
        CALL SCOPY ('       ',LEVEL3)
c--
c--     temporary diagnostics for debugging
c--
D       CALL PUTSTR(LTERM,' LOCATION: ','0')
D       CALL PUTSTR(LTERM,LEVEL1,' ')
D       CALL PUTSTR(LTERM,LEVEL2,' ')
D       CALL PUTSTR(LTERM,LEVEL3,' ')
c--
c--     store average values and flags in array locations
c--
        DO 30,J=1,NPL
        BOUDAT(J+40)=LBCF(J)     !LOC(41,42)
        BOUDAT(J+42)=LBPF(J)     !LOC(43,44)
        BOUDAT(J+46)=BCONAV(J)   !LOC(47,48)
30      BOUDAT(J+48)=BPRSAV(J)   !LOC(49,50)
c--
c--     store current system date and time
c--
        BOUDAT(36)=SYSTM1
        BOUDAT(37)=SYSDAT
c--
c--     boudat(60) is used in acc file for how many times thru
c--
        BOUDAT(60)=1.
c--
c--     keep current staus of plot and main pump
c--
        KSBP=KFBP
c--
c--     keep track of relay current status for future use
c--     also set pumps to be on to -1   use KPUMP flag
c--     this will turn current pumps on to off
c--
        DO 110,LP=1,NBPMP
        KPUMP(LP)=-1     !do not turn on if negative
110     KSBR(LP)=KFBR(LP)
c--
c--     check wind speed
```

```
c--
      IF (BOUDAT(25).LE.WINDMX) GOTO 18
      WRITE(LTERM,220) BOUDAT(25),WINDMX
      IWIND=1 !wind too strong
c--
c--   check temperature
c--
18    IF (BOUDAT(10).GE.TEMPMN) GOTO 19
      WRITE(LTERM,230) BOUDAT(10),TEMPMN
      ITEMP=1
19    IF (IWIND.EQ.0.AND.ITEMP.EQ.0) GOTO 35
c--
c--   log wind or temp out of range on terminal
c--
      IF (IWIND.EQ.1) WRITE (LTERM,*) 'Wind
      *speed exceeds maximum',BOUDAT(25)
      IF (ITEMP.EQ.1) WRITE (LTERM,*) 'Temperature below minimun',
      *BOUDAT(25)
c--
      RETURN
c--
c------------------------------------------------------------------------
c--
c--   setpoints are set in boudat.for
c--
c--   SET(1)=BOUDAT(RANL)
c--   SET(2)=BOUDAT(RANH)
c------------------------------------------------------------------------
35    SET(1)=BOUDAT(RANL)
      SET(2)=BOUDAT(RANH)
c--
c--   see if any pumps need to be turned on
c--
      LOCB=0  !flag to tell if any pumps need on
c--
      DO 11,LP=1,NBPMP
c--
c--
c--   check to see if we should really turn pumps on
c--   are probes bad?
c--
      DO 16,ITA1=1,NPL
      DO 16,ITA2=1,4
16    IF (LBCF(ITA1).NE.ITAB(ITA1,ITA2)) GOTO 14
      WRITE (LTERM,210) ITA1,LBCF,ITAB(ITAB1,ITAB2)
c--
c--   check ave against setpoint
c--
14    IF (SET(LP).GE.SETPTC(LP)) GOTO 11
c--
c--   inc counter for plot pump relay status -
c--
```

```
      LOCB=LOCB+1
c--
c--   pump to be turned on in is stored in KPUMP array
c--
      KPUMP(LP)=MBPMP(LP)
D     WRITE (LTERM,*) 'BOUMH2-pump status',LP,KSBR(LP),
D     *KFBR(LP),SET(LP),SETPTC(LP)
11    CONTINUE
      RETURN
c--------------------------------------------------------------------------
c--
c--   format statements
c--
c--------------------------------------------------------------------------
210   FORMAT (X,'PLOT ',I1,2X,'PROBE ERROR:',X,I1,'LOCATION: ',I1)
220   FORMAT (X,'WIND SPEED',F10.2.5X,'MAX WIND',F10.2)
230   FORMAT (X,'AIR TEMP',F10.2,,5X,'MIN TEMP',F10.2)
      END
      SUBROUTINE BOUMHT
c +-----------------------------------------------------------------+
c | Program :    BOULD.REL                                          |
c | Module  :    BOUMHT.FOR                                         |
c | Function:    CALCULATES AVERAGE TEMPERATURES FOR CONDUCTIVITY   |
c |              MEASUREMENTS. IF TEMPERATURE ERROR FLAG SET MAKES ALL |
c |              AVERAGE TEMPS = 20 c                               |
c |                                                                 |
c +-----------------------------------------------------------------+
c--
c--   data common block
c--
      COMMON/BOULD/BOUDAT(60),BOUA(25),BOUB(25),BDATA(60),BOUT(60),
     *NARRAY,BCONAV(2),BTMPAV(2),BPRSAV(2),NPL,NPR,NBT1,NBT2,NBC1,
     *NBC2,NBP1,NBP2,LBCF(2),LBTF(2),LBPF(2),SETPTC(2),LOCB,
     *KFBR(2),KFBP,KSBR(2),KSBP,KTMPF,KCONF,KPRSF,KINTF,KWNDF,KFFTF,
     *KPUMP(2)
c--
c--   data checking common block
c--
      COMMON/CONDCL/WINDMX,TEMPMN,BCONL,BCONH,BCONRF(3),BRI(3),
     *BVOLTS,BAI,BTMPRF,BCONIN,BAVETP(2),BTMPHZ,RANL,RANH,BFTDAC(4)
c--
c--   diagnostic block
c--
      LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
      INTEGER*4 I1AREA,I2AREA
      COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
     *LTERM,ITERM,ATIMES,BTIMES,IADPOK,IADPEK,IPHONE,ILDTR,IHDTR,
     *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
     *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
     *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
     *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
     *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
     *JBOVR,JBWAIT,JBSLEP
c--
c--   dimension for this section
c--    JTLOC= location to add to temperature readings to get proper
c--      value for testing to see if they are in range
c--
      DIMENSION JTLOC(6)
      DATA JTLOC/1,2,2,3,3,1/
c--
c--   set bg diagnostics
c--
      CALL IPOKEB (JBDS1,JBMAH+2)
c--
c--   timing failure traceback
c--
      CALL SCOPY (' BOUMHT',LEVEL2)
      CALL SCOPY ('       ',LEVEL3)
c--
```

```
c--     tempory diagnostics for debugging
c--
D       CALL PUTSTR(LTERM,' LOCATION: ','0')
D       CALL PUTSTR(LTERM,LEVEL1,' ')
D       CALL PUTSTR(LTERM,LEVEL2,' ')
D       CALL PUTSTR(LTERM,LEVEL3,' ')
c--------------------------------------------------------------------------
c--
c--      calculate average for temperatures
c--------------------------------------------------------------------------
c--
c--     init flags for out of range
c--
        DO 9,JH=1,NPL
        BTMPAV(JH)=0.
9       LBTF(JH)=0
c--
c--     check to see if temp error flag set
c--     on board failure make  ave temp =20C
c--
        IF (KTMPF.LE.0) GOTO 8
        WRITE (LTERM,*) 'Temperature module problem',SYSDAT,SYSTM1
c--
c--     tell which error
c--
        IF (KTMPF.GE.2) GOTO 7
        WRITE (LTERM,*) 'Deviation of reference',BOUDAT(1)
        GOTO 8
7       IF (KTMPF.EQ.2) WRITE (LTERM,*) 'All probes 0. - board failure'
        IF (KTMPF.EQ.3) WRITE (LTERM,*) 'Lockup of temperature probe'
        IF (KTMPF.EQ.4) WRITE (LTERM,*) 'Possible lockup of
        *temperature module. Ref is',BOUDAT(1)
        GOTO 201
c--
c--     sum numbers
c--
c--
c--     check to see if any out of range
c--     do this by number of plots and number off probes
c--
c--
8       DO 10 J=1,NPL   !PLOTS
c--
c--     IZ. is where in the data array to store number
c--     this is done this way so if more data and/or sensors are added
c--     the array locations would automatically be corrected
c--
        IF (J.EQ.1) IZT=NBT1-1
        IF (J.EQ.2) IZT=NBT2-1
c--
c--     check to see if readings fall within the specified range
c--
```

```
c--     the number 7 is used for the temperature range because
c--             5 % range = 0.05 *100 degrees (-50 to 50 degrees c)= 5 des
c--             probe accurate to 1 degree c                         1 des
c--             an extra degree for drift                            1 des
c--                                                                 ---
c--                                                                  7 des
c--
c--
c--
        DO 10 I=1,5,2
D       WRITE (LTERM,*) 'BOUDAT(JTLOC(I)+IZT)',BOUDAT(JTLOC(I)+IZT)
D       WRITE (LTERM,*) 'BOUDAT(JTLOC(I+1)+IZT)-7',BOUDAT(JTLOC(I+1)+IZT).
D       WRITE (LTERM,*) 'BOUDAT(JTLOC(I+1)+IZT)+7',BOUDAT(JTLOC(I+1)+IZT).
        IF ((BOUDAT(JTLOC(I)+IZT).LT.BOUDAT(JTLOC(I+1)+IZT)-7.)
            .OR.(BOUDAT(JTLOC(I)+IZT).GT.BOUDAT(JTLOC(I+1)+IZT)+7.))
                GOTO 10
c--
c--     temperatures are calculated by using a weighted average
c--     if both readings are in range they are added and then averaged
c--     LBTF is used temporarily as a counter to tell how many probes are
c--     in range
c--
        LBTF(J)=LBTF(J)+2
        BTMPAV(J)=BTMPAV(J)+BOUDAT(JTLOC(I)+IZT)+BOUDAT(JTLOC(I+1)+IZT)
10      CONTINUE
c--
c--
c--     now that we have the sums lets do the averages
c--
        DO 11,I=1,NPL
        IF (LBTF(I).EQ.0) GOTO 11
        BTMPAV(I)=BTMPAV(I)/LBTF(I)
11      CONTINUE
c------------------------------------------------------------
c--
c--     check to see if flag set, if set recalculate
c--
c----------------------------------------------------------
        DO 13,IT=1,NPL
        IPROBE=IT
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c--
c--
c--     calculate number out of range = to number probes -number in range
c--
        LBTF(IT)=NPR-(LBTF(IT)/2)
20      IF (LBTF(IT).EQ.0) GOTO 13
c--
c--
c--     if ) 2 branch --- tell that there is a problem!!
c--
        WRITE (LTERM,211) IPROBE,LBTF(IT)
        IF (LBTF(IT).LT.3) GOTO 13
c--
c--     too many probes deviated make = to setpoint
c--
200     BTMPAV(IT)=BAVETP(IT)
13      CONTINUE
c--
c--     store average values and flags in array locations
c--
204     DO 30,J=1,NPL
        BOUDAT(J+38)=LBTF(J)     !LOC(39,40)
30      BOUDAT(J+44)=BTMPAV(J)   !LOC(45,46)
        RETURN
c--
c--     temp flag error
c--
201     DO 202,I=1,NPL
        LBTF(I)=256      !error with module
202     BTMPAV(I)=BAVETP(I)
        GOTO 204
c-----------------------------------------------------------------------
c--     format statements
c-----------------------------------------------------------------------
211     FORMAT (' PLOT',I2,' HAS',I2,' TEMPERATURE PROBES
        *OUT OF RANGE')
        END
```

```
      SUBROUTINE BOUOSC
c +-----------------------------------------------------------------------+
c | Program :    BOULD.REL                                                 |
c | Module  :    BOUOSC.FOR                                                |
c | Function:    READS AND VERIFY OPERATION OF OSCILLATOR                  |
c |                                                                        |
c +-----------------------------------------------------------------------+
c--
c--     data common block
c--
      COMMON/BOULD/BOUDAT(60),BOUA(25),BOUB(25),BDATA(60),BOUT(60),
     *NARRAY,BCONAV(2),BTMPAV(2),BPRSAV(2),NPL,NPR,NBT1,NBT2,NBC1,
     *NBC2,NBP1,NBP2,LBCF(2),LBTF(2),LBPF(2),SETPTC(2),LOCB,
     *KFBR(2),KFBP,KSBR(2),KSBP,KTMPF,KCONF,KPRSF,KINTF,KWNDF,KFFTF,
     *KPUMP(2)
c--
c--     channel switching common block
c--
      COMMON/CHAN/IBCONB,IBCOO,IBFCI,IBVFI,IBVFO,IBTCM,IBREFR,
     *IBWS,IBMPR,IBPPR,IBPNIC,MBCON(7),NBCON,MBTMP(10),
     *NBTMP,MBPMP(2),NBPMP,MBPRS(7),NBPRS,MBFFT(2),NBFFT,
     *MBCONR(3),NBCONR
c--
c--     data checking common block
c--
      COMMON/CONDCL/WINDMX,TEMPMN,BCONL,BCONH,BCONRF(3),BRI(3),
     *BVOLTS,BAI,BTMPRF,BCONIN,BAVETP(2),BTMPHZ,RANL,RANH,BFTDAC(4)
c--
c--     control character common block
c--
      COMMON /CONTRL/LA,LB,LC,LR,LDLE,LQU,LLAS,LTACT,LLACT,LTAC,
     *LLAC,LI,LF,LCR,LLF,ITRH1,ITRM1,ITRS1,ITRT1,ITRT2,ITRT3
c--
c--     diagnostic block
c--
      LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
      INTEGER*4 I1AREA,I2AREA
      COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
     *LTERM,ITERM,ATIMES,BTIMES,IADPOK,IADPEK,IPHONE,ILDTR,IHDTR,
     *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
     *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
     *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
     *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
     *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
     *JBOVR,JBWAIT,JBSLEP
c--
c--     -----------------------------------------------------------------
c--
c--     initialize error flag
c--
c--
c--     bg program diagnostics
```

```
C--
      CALL IPOKEB(JBDS1,JBCON+1)
C--
C--   timing failure traceback
C--
      CALL SCOPY (' BOUOSC',LEVEL2)
      CALL SCOPY ('       ',LEVEL3)
C--
C--   temporary diagnostics for debugging
C--
D     CALL PUTSTR(LTERM,' LOCATION: ','0')
D     CALL PUTSTR(LTERM,LEVEL1,' ')
D     CALL PUTSTR(LTERM,LEVEL2,' ')
D     CALL PUTSTR(LTERM,LEVEL3,' ')
C--
C--
      KCONF=0 !init error flag
      KPRSF=0 !init error flag
      LOCB=11 !array locator for boudat array
      ITRY=0  !init try counter for valid reading
C--
C--   check oscillator
C--
100   CALL RLLTI(MFLAG)       !sets conditions so channels can be closed
      IF (MFLAG.EQ.1) GO TO 100
C--
C--   close voltage to frequency convertor input and test error flag
C--
      CALL BOUCHN (IBFCI,MFLAG)
      IF (MFLAG.EQ.1) GO TO 100
C--
C--   close conductance oscillator output
C--
      CALL BOUCHN (IBCOO,MFLAG)
      IF (MFLAG.EQ.1) GO TO 100
C--
C--   close channels
C--
      CALL RET(MFALG) !tells scanner all commands have been sent
      IF (MFLAG.EQ.1) GO TO 100
C--
      CALL RTLLI(MFLAG)       !sets conditions so data can be received
      IF (MFLAG.EQ.1) GO TO 100
C--
      CALL WRITE(BOUDAT(LOCB),MFLAG)  !converts data to real*4 number
      IF (MFLAG.EQ.1) GO TO 100
C--
C--   convert to hz
C--
D     WRITE (LTERM,*) 'BOUOSC READING',BOUDAT(LOCB)
      BOUDAT(LOCB)=BOUDAT(LOCB)*1000.
C--
```

```
c--
c--     reading too high ????
c--
120     IF (BOUDAT(LOCB).GT.900.
        * .AND. BOUDAT(LOCB).LT.2200.)
        *   GOTO 109
        ITRY=ITRY+1
        TYPE *,'OSCILLATOR OUT OF RANGE ',BOUDAT(LOCB)
        IF (ITRY.LE.5) GOTO 100
        GOTO 140
c--
c--     oscillator reads square wave picking up both positive and
c--     negative pulses on occasion.
c--     this causes the reading to be twice what it should be.
c--
109     IF (BOUDAT(LOCB).GT.1800.AND.
        *  BOUDAT(LOCB).LT.2200.) BOUDAT(LOCB)=BOUDAT(LOCB)/2.
c--
c---    check osciilator range if error set flag
c--      check to see if in 50% on each side of reference
c--       reference =  1kHz or 1000Hz
c--
        RETURN
c--
c--     error reporting
c--
140     IPOK=(IPEEKB(JBDERR))+JBECON
        CALL IPOKEB(JBDERR,IPOK)
        KCONF=1
        RETURN
        END
```

```
      SUBROUTINE BOUPDC(IEQUIP)
c +--------------------------------------------------------------------+
c | Program :    BOULD.REL                                             |
c | Module  :    BOUPDC.FOR                                            |
c | Function:    READS dac TO CHECK SYNC AND IF NEED REESTABLISHES SYNC |
c |                                                                    |
c +--------------------------------------------------------------------+
c |  NOTE:                                                             |
c |         Bit codes are explained in BOUDAT.FOR                      |
c |                                                                    |
c |     error codes                                                    |
c |           0 - normal                                               |
c |           3 - sync problem                                         |
c |                                                                    |
c +--------------------------------------------------------------------+
c--
c--   data common block
c--
      COMMON/BOULD/BOUDAT(60),BOUA(25),BOUB(25),BDATA(60),BOUT(60),
      *NARRAY,BCONAV(2),BTMPAV(2),BPRSAV(2),NPL,NPR,NBT1,NBT2,NBC1,
      *NBC2,NBP1,NBP2,LBCF(2),LBTF(2),LBPF(2),SETPTC(2),LOCB,
      *KFBR(2),KFBP,KSBR(2),KSBP,KTMPF,KCONF,KPRSF,KINTF,KWNDF,KFFTF,
      *KPUMP(2)
c--
c--   channel switching common block
c--
      COMMON/CHAN/IBCONB,IBCOO,IBFCI,IBVFI,IBVFO,IBTCM,IBREFR,
      *IBWS,IBMPR,IBPPR,IBPNIC,MBCON(7),NBCON,MBTMP(10),
      *NBTMP,MBPMP(2),NBPMP,MBPRS(7),NBPRS,MBFFT(2),NBFFT,
      *MBCONR(3),NBCONR
c--
c--   data checking common block
c--
      COMMON/CONDCL/WINDMX,TEMPMN,BCONL,BCONH,BCONRF(3),BRI(3),
      *BVOLTS,BAI,BTMPRF,BCONIN,BAVETP(2),BTMPHZ,RANL,RANH,BFTDAC(4)
c--
c--   control character common block
c--
      COMMON /CONTRL/LA,LB,LC,LR,LDLE,LQU,LLAS,LTACT,LLACT,LTAC,
      *LLAC,LI,LF,LCR,LLF,ITRH1,ITRM1,ITRS1,ITRT1,ITRT2,ITRT3
c--
c--   diagnostic block
c--
      LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
      INTEGER*4 I1AREA,I2AREA
      COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
      *LTERM,ITERM,ATIMES,BTIMES,IADPOK,IADPEK,IPHONE,ILDTR,IHDTR,
      *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
      *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
      *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
      *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
      *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
        *JBOVR,JBWAIT,JBSLEP
c-----------------------------------------------------------------------
c--
c--     dimensioning and common for this routine
c--
c-----------------------------------------------------------------------
c--
c--
c--     bits common
c--
        COMMON/BITS/IBIT(16),IWORD
c--
c--
        REAL*4 LABEL(4,3)
        DATA LABEL/'Plot','Plot','Plot','Prim',
        *' 1 r',' 2 r',' pum','ary ',
        *'elay','elay','p   ','pump'/
c--
        DATA IQTIM1/10/IQTIM2/20/
c--
        IEQUIP=0        !init error flag
        ISYNC=0         !test for sync of plot pump
c--
c--     diagnostic information
c--
c--     bg diagnostics
c--
        CALL IPOKEB(JBDS1,JBPMP+1)
c--
c--     timing failure traceback diagnostics
c--
        CALL SCOPY (' BOUPDC',LEVEL2)
        CALL SCOPY ('       ',LEVEL3)
c--
c--     temporary diagnostics
c--
D       CALL PUTSTR(LTERM,' LOCATION: ','0')
D       CALL PUTSTR(LTERM,LEVEL1,' ')
D       CALL PUTSTR(LTERM,LEVEL2,' ')
D       CALL PUTSTR(LTERM,LEVEL3,' ')
c--
c-------------------------------------------------------------------------
        LOCB=27
19      ITRY=0  !init zero check
c--
c--     start control sequence
c--
20      CALL RLLTI(MFLAG)       !set conditions so channels can be closed
        IF (MFLAG.EQ.1) GO TO 20
c--
c--     close frequency counter input
c--
```

```
      CALL BOUCHN (IBFCI,MFLAG)
      IF (MFLAG.EQ.1) GO TO 20
c--
c--   close voltage frequency input
c--
      CALL BOUCHN (IBVFI,MFLAG)
      IF (MFLAG.EQ.1) GO TO 20
c--
c--   close voltage/frequency output
c--
      CALL BOUCHN (IBVFO,MFLAG)
      IF (MFLAG.EQ.1) GO TO 20
c--
c--   close DAC converter
c--
      CALL BOUCHN (IBPNIC,MFLAG)
      IF (MFLAG.EQ.1) GO TO 20
c--
c--   finish control sequence
c--
      CALL RET(MFLAG) !tells scanner all commands have been sent
      IF (MFLAG.EQ.1) GO TO 20
c--
      CALL RTLLI(MFLAG)        !sets conditions so data can be received
      IF (MFLAG.EQ.1) GO TO 20
c--
      CALL WRITE(BOUDAT(LOCB),MFLAG)  !converts data to real*4 number
      IF (MFLAG.EQ.1) GO TO 20
c--
c--   check for zero condition
c--   a zero may not be a zero but a bad reading ckeck and see
c--
      ITRY=ITRY+1
      IF (ITRY.LE.5 .AND.BOUDAT(LOCB) .EQ. 0.) GOTO 20
c--
c--   check to see if in proper range
c--   if not go back and get reading again
c--
      IF (BOUDAT(LOCB) .LT. 0.0 .OR.
      * BOUDAT(LOCB) .GT.100.0) GOTO 19
      IF (BOUDAT(LOCB) .EQ. 0.0) GOTO 999
c--
c--   convert reading
c--
c--
      CALC = (BFTDAC(1)-BOUDAT(LOCB))/BFTDAC(2)
      IWORD=CALC
      ZHOLD=DIM(CALC,FLOAT(IWORD))
      IF (ZHOLD.GE.0.5) IWORD=IWORD+1
c--
D     WRITE (LTERM,*) 'BOUPDC AFTER CALC',BOUDAT(LOCB),BFTDAC(1),
D     *BFTDAC(2),IWORD
```

```
        CALL BOUBIT
D       WRITE(LTERM,800) KFBP,(IBIT(K),K=16,1,-1),BOUDAT(LOCB)
c--
c--     out put results for debug
c--
c--     check to see if any problems with the readings
c--
c--     is pump status correct
c--
        IF (KFBP .EQ. IBIT(8)
        * .AND. KFBP .EQ. IBIT(7))
        *   GOTO 900
c--
c--     pumps out of sync
c--
        WRITE (LTERM,*) 'BOUPDC - Pumps out of sync'
        WRITE(LTERM,800) KFBP,(IBIT(K),K=16,1,-1),BOUDAT(LOCB)
c--
c--     section to flip pump status
c--
c--
c--     inc sync counter
c--
37      ISYNC=ISYNC+1
c--
c--     if < IQTIM1 go ahead and change with no warning
c--
        IF (ISYNC.LT.IQTIM1) GOTO 36
c--
c--     give sync warning if more then IQTIM1 times      set error flag ao
c--
        WRITE (LTERM,*) 'BOUPDC - Sync problem times ',ISYNC
        IEQUIP=3
c--
c--     set dac reading to negative to show error
c--
        BOUDAT(27)= (-1*BOUDAT(LOCB))    !make dac - to show sync error
c--
c--     if more then IQTIM2 time log error and return try to take care ofs
c--
        IF (ISYNC.GT.IQTIM2) goto 950
c--
c--     was primary pump the one out of sync????
c--
36      IF (IBIT(8) .EQ. KFBP) GOTO 35
        ILOC=IBMPR
        ASSIGN 35 TO IBR
        GOTO 201
c--
c--     was plot pump the one out of sync?????
c--
35      IF (IBIT(7) .EQ. KFBP) GOTO 19
```

```
        ILOC=IBPPR
        ASSIGN 19 TO IBR
        GOTO 201
c--
c--     check to see if any manual operation!!
c--
900     DO 901 I=3,6
901     IF (IBIT(I).EQ.1) GOTO 905
        GOTO 30
c--
c--     some pumps on tell which ones
c--
905     WRITE (LTERM,*) 'BOUPDC - Manual operation status'
        DO 907 I=1,4
907     IF (IBIT(I+2).EQ.1) WRITE (LTERM,908) (LABEL(I,J),J=1,3)
c--
c--     normal return
c--
30      IEQUIP=0
        RETURN
c--
c--     channel switching routine
c--
201     CALL RLLTI(MFLAG)
        IF (MFLAG.EQ.1) GOTO 201
        CALL BOUCHN(ILOC,MFLAG)
        IF (MFLAG.EQ.1) GOTO 201
        CALL RET (MFLAG)          !now close channel
        IF (MFLAG.EQ.1) GOTO 201
        CALL ISLEEP(0,0,0,5)      !give time to stablize if needed
        WRITE (LTERM,*) 'BOUPDC - CHANNEL TOGGLED',ILOC
        GOTO IBR
c--
c--     dac error
c--
999     WRITE (LTERM,*) 'BOUPDC - DAC ERROR',BOUDAT(LOCB)
        WRITE (LTERM,*) 'BOUPDC - DAC SHOULD NOT BE ZERO.'
        RETURN
c--
c--     error with sync
c--
950     WRITE (LTERM,*) 'BOUPDC error in sync timing - trying to
        *turn pumps off'
c--
c--     set pumps sattus to off (0)
c--
        ISYNC=0
        KFBP=0
        GOTO 19
c+-------------------------------------------------------------+
c|                                                             |
c|                    format statements                        |
c|                                                                           |
c+---------------------------------------------------------------------------+
800     FORMAT (X,'BOUPDC Pump flag',3x,I3,3X,'BITS',3X,16I2,
        * X,'DAC',F5.2)
908     FORMAT (X,3A4)
D9000   FORMAT (X,5X,'DAC',E12.5,5X,I5,F10.0,5X,F10.0,5X,10A1)
        END
```

```
      SUBROUTINE BOUPDF(IEQUIP)
c +----------------------------------------------------------------------+
c | Program :     BOULD.REL                                              |
c | Module  :     BOUPDF.FOR                                             |
c | Function:     READS dac TO CHECK SYNC AND IF NEED REESTABLISHES SYNC |
c |               THIS ROUTINE IS SIMILAR TO boupdc.for EXCEPT IT        |
c |               CONTINUOUSLY TRIES TO REESTABLISH SYNC AND HAS GREATER |
c |               TIMING DELAYS.IF SYNC CAN NOT BE REESTABLISHED IT TRIES |
c |               OPEN ALL PUMPS                                         |
c |                                                                      |
c +----------------------------------------------------------------------+
c--
c--     data common block
c--
      COMMON/BOULD/BOUDAT(60),BOUA(25),BOUB(25),BDATA(60),BOUT(60),
     *NARRAY,BCONAV(2),BTMPAV(2),BPRSAV(2),NPL,NPR,NBT1,NBT2,NBC1,
     *NBC2,NBP1,NBP2,LBCF(2),LBTF(2),LBPF(2),SETPTC(2),LOCB,
     *KFBR(2),KFBP,KSBR(2),KSBP,KTMPF,KCONF,KPRSF,KINTF,KWNDF,KFFTF,
     *KPUMP(2)
c--
c--     channel switching common block
c--
      COMMON/CHAN/IBCONB,IBCOO,IBFCI,IBVFI,IBVFO,IBTCM,IBREFR,
     *IBWS,IBMPR,IBPPR,IBPNIC,MBCON(7),NBCON,MBTMP(10),
     *NBTMP,MBPMP(2),NBPMP,MBPRS(7),NBPRS,MBFFT(2),NBFFT,
     *MBCONR(3),NBCONR
c--
c--     data checking common block
c--
      COMMON/CONDCL/WINDMX,TEMPMN,BCONL,BCONH,BCONRF(3),BRI(3),
     *BVOLTS,BAI,BTMPRF,BCONIN,BAVETP(2),BTMPHZ,RANL,RANH,BFTDAC(4)
c--
c--     control character common block
c--
      COMMON /CONTRL/LA,LB,LC,LR,LDLE,LQU,LLAS,LTACT,LLACT,LTAC,
     *LLAC,LI,LF,LCR,LLF,ITRH1,ITRM1,ITRS1,ITRT1,ITRT2,ITRT3
c--
c--     diagnostic block
c--
      LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
      INTEGER*4 I1AREA,I2AREA
      COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
     *LTERM,ITERM,ATIMES,BTIMES,IADPOK,IADPEK,IPHONE,ILDTR,IHDTR,
     *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
     *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
     *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
     *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
     *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
     *JBOVR,JBWAIT,JBSLEP
c----------------------------------------------------------------------
c--
c--     dimensioning and common for this routine
```

```
c--
c------------------------------------------------------------------------
c--
c--     common block for boubits
c--
        COMMON/BITS/IBIT(16),IWORD
        REAL*4 LABEL(4,3)
        DATA LABEL/'Prim','Plot','Plot','Plot',
        *'ary ',' pum',' 1 r',' 2 r',
        *'pump','p   ','elay','elay'/
c--
        IEQUIP=0        !init error flag
        ISYNC=0 !test for sync of plot pump
c--
c--     diagnostic information
c--
c--     bg diagnostics
c--
        CALL IPOKEB(JBDS1,JBPMP+1)
c--
c--     timing failure traceback diagnostics
c--
        CALL SCOPY (' BOUPDC',LEVEL2)
        CALL SCOPY ('       ',LEVEL3)
c--
c--     temporary diagnostics
c--
D       CALL PUTSTR(LTERM,' LOCATION: ','0')
D       CALL PUTSTR(LTERM,LEVEL1,' ')
D       CALL PUTSTR(LTERM,LEVEL2,' ')
D       CALL PUTSTR(LTERM,LEVEL3,' ')
c--
c------------------------------------------------------------------------
19      ITRY=0  !init zero check
c--
c--     start control sequence
c--
20      CALL RLLTI(MFLAG)       !set conditions so channels can be closed
        IF (MFLAG.EQ.1) GO TO 20
c--
c--     close frequency counter input
c--
        CALL BOUCHN (IBFCI,MFLAG)
        IF (MFLAG.EQ.1) GO TO 20
c--
c--     close voltage frequency input
c--
        CALL BOUCHN (IBVFI,MFLAG)
        IF (MFLAG.EQ.1) GO TO 20
c--
c--     close voltage/frequency output
c--
```

```
      CALL BOUCHN (IBVFO,MFLAG)
      IF (MFLAG.EQ.1) GO TO 20
c--
c--   close DAC converter
c--
      CALL BOUCHN (IBPNIC,MFLAG)
      IF (MFLAG.EQ.1) GO TO 20
c--
c--   finish control sequence
c--
      CALL RET(MFLAG) !tells scanner all commands have been sent
      IF (MFLAG.EQ.1) GO TO 20
c--
      CALL RTLLI(MFLAG)       !sets conditions so data can be received
      IF (MFLAG.EQ.1) GO TO 20
c--
      CALL WRITE(TEST,MFLAG)  !converts data to real*4 number
      IF (MFLAG.EQ.1) GO TO 20
c--
c--   check for zero condition
c--   a zero may not be a zero but a bad reading ckeck and see
c--
      ITRY=ITRY+1
      IF (ITRY.LE.5 .AND.TEST .EQ. 0.) GOTO 20
c--
c--   convert form kHz to Hz (/1000.)
c--
      IF (TEST.EQ.0) GOTO 333
      TEST=1./TEST
333   TEST=TEST/1000.
c--
c--   check to see if in proper range
c--   if not go back and get reading again
c--
      IF (TEST .LT. 0.0 .OR. TEST .GT.100.0) GOTO 19
      IF (TEST .EQ. 0.0) GOTO 999
c--
c--   convert reading
c--
      IWORD = (BFTDAC(1)-TEST)/BFTDAC(2)
      CALL BOUBIT
c--
c--   out put results for debug
c--
D     WRITE(LTERM,9000) TEST,IWORD,SYSTM1,SYSDAT,LEVEL2
c--
c--   check to see if any problems with the readings
c--
c--   is pump status correct
c--
      IF (KFBP .EQ. BIT(8)
      * .AND. KFBP .EQ. BIT(7))
```

```
        *   GOTO 900
c--
c--     pumps out of sync
c--
        TYPE *,'Pumps out of sync'
c--
c--     section to flip pump status
c--
c--
c--     inc sync counter
c--
37      ISYNC=ISYNC+1
c--
c--     if (10 go ahead and change with no warning
c--
        IF (ISYNC.LT.10) GOTO 36
c--
c--     give sync warning if more then 10 times      set error flag also
c--
        TYPE *,'Sync problem times ',ISYNC
        IEQUIP=3
c--
c--     set dac reading to negative to show error
c--
        BOUDAT(27)= (-1*IWORD)  !makc dac - to show sync error
c--
c--     if more thgen 20 time log error and return try to take care of ths
c--     in another routine
c--
        IF (ISYNC.GT.20) RETURN
c--
c--     was primary pump the one out of sync????
c--
36      IF (IBIT(8) .EQ. KFBP) GOTO 35
        ILOC=IBMPR
        ASSIGN 35 TO IBR
        GOTO 201
c--
c--     was plot pump the one out of sync?????
c--
35      IF (IBIT(7) .EQ. KFBP) GOTO 19
        ILOC=IBPPR
        ASSIGN 19 TO IBR
        GOTO 201
c--
c--     check to see if any manual operation!!
c--
900     DO 901 I=3,6
901     IF (IBIT(I).EQ.1) GOTO 905
        GOTO 30
c--
c--     some pumps on tell which ones
```

```
c--
905     TYPE *,'Manual operation status'
        DO 907 I=1,4
907     IF (IBIT(I+2).EQ.1) TYPE 908,(LABEL(I,J),J=1,3)
c--
c--     normal return
c--
30      IEQUIP=0
        BOUDAT(27)=IWORD
        RETURN
c--
c--     channel switching routine
c--
201     CALL RLLTI(MFLAG)
        IF (MFLAG.EQ.1) GOTO 201
        CALL BOUCHN(ILOC,MFLAG)
        IF (MFLAG.EQ.1) GOTO 201
        CALL RET (MFLAG)          !now close channel
        IF (MFLAG.EQ.1) GOTO 201
        CALL ISLEEP(0,0,0,5)      !give time to stablize if needed
        WRITE (LTERM,*) 'CHANNEL TOGGLED',ILOC
        GOTO IBR
999     TYPE *,'DAC ERROR',TEST
        TYPE *,'DAC SHOULD NOT BE ZERO.'
        RETURN
c+--------------------------------------------------------------------+
c|                                                                    |
c|                        format statements                           |
c|                                                                    |
c+--------------------------------------------------------------------+
908     FORMAT (X,3A4)
D9000   FORMAT (X,5X,'DAC',2(E12.5,5X),F10.0,5X,F10.0,5X,10A1)
        END
```

```
      SUBROUTINE BOUPFT(IEQFFT)
c +-------------------------------------------------------------------+
c | Program :    BOULD.REL                                            |
c | Module  :    BOUPFT.FOR                                           |
c | Function:    READS FLOW TRANSDUCER TO VERIFY PLOT RELAYS WERE TURNED |
c |                                                                   |
c +-------------------------------------------------------------------+
c | NOTE:                                                             |
c |    uses KFBP  and KFBR(Z) flag to tell if relays are to be on     |
c +-------------------------------------------------------------------+
c--
c--   data common block
c--
      COMMON/BOULD/BOUDAT(60),BOUA(25),BOUB(25),BDATA(60),BOUT(60),
     *NARRAY,BCONAV(2),BTMPAV(2),BPRSAV(2),NPL,NPR,NBT1,NBT2,NBC1,
     *NBC2,NBP1,NBP2,LBCF(2),LBTF(2),LBPF(2),SETPTC(2),LOCB,
     *KFBR(2),KFBP,KSBR(2),KSBP,KTMPF,KCONF,KPRSF,KINTF,KWNDF,KFFTF,
     *KPUMP(2)
c--
c--   channel switching common block
c--
      COMMON/CHAN/IBCONB,IBCOO,IBFCI,IBVFI,IBVFO,IBTCM,IBREFR,
     *IBWS,IBMPR,IBPPR,IBPNIC,MBCON(7),NBCON,MBTMP(10),
     *NBTMP,MBPMP(2),NBPMP,MBPRS(7),NBPRS,MBFFT(2),NBFFT,
     *MBCONR(3),NBCONR
c--
c--   data checking common block
c--
      COMMON/CONDCL/WINDMX,TEMPMN,BCONL,BCONH,BCONRF(3),BRI(3),
     *BVOLTS,BAI,BTMPRF,BCONIN,BAVETP(2),BTMPHZ,RANL,RANH,BFTDAC(4)
c--
c--   control character common block
c--
      COMMON /CONTRL/LA,LB,LC,LR,LDLE,LQU,LLAS,LTACT,LLACT,LTAC,
     *LLAC,LI,LF,LCR,LLF,ITRH1,ITRM1,ITRS1,ITRT1,ITRT2,ITRT3
c--
c--   diagnostic block
c--
      LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
      INTEGER*4 I1AREA,I2AREA
      COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
     *LTERM,ITERM,ATIMES,BTIMES,IADPOK,IADPEK,IPHONE,ILDTR,IHDTR,
     *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
     *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
     *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
     *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
     *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
     *JBOVR,JBWAIT,JBSLEP
c--
c--   diagnostic information
c--
c--   bg diagnostics
```

```
c--
        CALL IPOKEB(JBDS1,JBPMP+2)
c--
c--     timing failure traceback diagnostics
c--
        CALL SCOPY (' BOUPFT',LEVEL2)
        CALL SCOPY ('       ',LEVEL3)
c--
c--     temporary diagnostics
c--
D       CALL PUTSTR(LTERM,' LOCATION: ','0')
D       CALL PUTSTR(LTERM,LEVEL1,' ')
D       CALL PUTSTR(LTERM,LEVEL2,' ')
D       CALL PUTSTR(LTERM,LEVEL3,' ')
c--
c-----------------------------------------------------------------------
c--
        DO 30,L=1,NBFFT
        ITRY=0  !init zero check
c--
c--     are pumps supposed to be on ? if not skip
c--
        IF (KFBR(L).EQ.0) GOTO 30
        ILOC=MBPMP(L)
c--
c--     start control sequence
c--
20      CALL ISLEEP(0,0,1,0)      !give time for valves
        CALL RLLTI(MFLAG)         !set conditions so channels can be closed
        IF (MFLAG.EQ.1) GO TO 20
c--
c--     close frequency counter input
c--
        CALL BOUCHN (IBFCI,MFLAG)
        IF (MFLAG.EQ.1) GO TO 20
c--
c--     close frequency transducer for plot
c--
        CALL BOUCHN (MBFFT(L),MFLAG)
        IF (MFLAG.EQ.1) GO TO 20
c--
c--     finish control sequence
c--
        CALL RET(MFLAG) !tells scanner all commands have been sent
        IF (MFLAG.EQ.1) GO TO 20
c--
        CALL RTLLI(MFLAG)         !sets conditions so data can be received
        IF (MFLAG.EQ.1) GO TO 20
c--
        CALL WRITE(TEST,MFLAG)  !converts data to real*4 number
        IF (MFLAG.EQ.1) GO TO 20
c--
```

```
c--     check for zero condition
c--     a zero may not be a zero but a bad reading ckeck and see
c--     in this section no zero's should be read therefore an error
c--     has occurred
c--
        ITRY=ITRY+1
        IF (ITRY.LE.5 .AND.TEST .EQ. 0.) GOTO 20
c--
c--     check for reading if not 0.relays on properly
c--     no conversion to freq.     reading in time period as
c--     we do not care what it is only that it is > 0.
c--     a 0. means the relay did not turn on properly.
c--
        IF (TEST.NE.0) GOTO 30
c--
c--     channel switching routine
c--
        WRITE (LTERM,*) 'ERROR ON RELAY CLOSING - CHANNEL',ILOC,SYSTM1,SYT
201     CALL RLLTI(MFLAG)
        IF (MFLAG.EQ.1) GOTO 201
        CALL BOUCHN(ILOC,MFLAG)
        IF (MFLAG.EQ.1) GOTO 201
        CALL RET          !close channels
        ITRY=0  !init time to try 0 reading
        GOTO 20 !recheck reading
c--
c--
c--
30      CONTINUE
c--
        RETURN
c--
c--     error section
c--
150     IEQFFT=1
        WRITE (LTERM,*) 'FLOW RELAY PROBLEM',ILOC,SYSTM1,SYSDAT
        RETURN
        END
```

FF

```
        SUBROUTINE BOUPKC
c +-------------------------------------------------------------------+
c | Program :    BOULD.REL                                            |
c | Module  :    BOUPKC.FOR                                           |
c | Function:    RESET DIAGNOSTIC BITS TO 0                           |
c |                                                                   |
c +-------------------------------------------------------------------+
c--
c--     diagnostic block
c--
        LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
        INTEGER*4 I1AREA,I2AREA
        COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
     *LTERM, ITERM, ATIMES, BTIMES, IADPOK, IADPEK, IPHONE, ILDTR, IHDTR,
     *I1AREA(4), I2AREA(4), SYSTM1, SYSDAT, JBDIO, JBDPGM, JBDS1, JBDS2,
     *JBDCH1, JBDCH2, JBDCTL, JBDERR, JBDAL, JBTMP, JBWND, JBCON, JBPRS, JBFFT,
     *JBERR, JBMAH, JBPMP, JBSTA, JBTIM, JBDK1, JBSLP, JBPKC, JBINT,
     *JBRTL, JBCHN, JBRLL, JBWRT, JBRET, JBRED, JBBIT, JBFAL, JBDISK, JBTT,
     *JBECON, JBEINT, JBETMP, JBEFFT, JBEDAL, JBEHND, JBEADD, JBESHT, JBRUN,
     *JBOVR, JBWAIT, JBSLEP
c--
      CALL IPOKEB (JBDS1, JBPKC)
c--
c--   byte JBDIO used for output determination and FG/BG
c--   interaction so it does not get reset
c--
c--   byte JBDPGM is used to see if the fg program
c--   is still running and does not get reset
c--
c--   byte defined by JBDCH2 is the override byte
c--   this allows the program to execute even though all error flags tel
c--   it to enter the shut down mode
c--
      CALL IPOKEB (JBDS1,0)
      CALL IPOKEB (JBDS2,0)
      CALL IPOKEB (JBDCH1,0)
      CALL IPOKEB (JBDCTL,0)
      CALL IPOKEB (JBDERR,0)
      RETURN
      END
```

```
        SUBROUTINE BOUPMP
c +----------------------------------------------------------------------+
c | Program :    BOULD.REL                                               |
c | Module  :    BOUPMP.FOR                                              |
c | Function:    TURNS PUMPS ON                                          |
c |                                                                      |
c +----------------------------------------------------------------------+
c--
c--     data common block
c--
        COMMON/BOULD/BOUDAT(60),BOUA(25),BOUB(25),BDATA(60),BOUT(60),
        *NARRAY,BCONAV(2),BTMPAV(2),BPRSAV(2),NPL,NPR,NBT1,NBT2,NBC1,
        *NBC2,NBP1,NBP2,LBCF(2),LBTF(2),LBPF(2),SETPTC(2),LOCB,
        *KFBR(2),KFBP,KSBR(2),KSBP,KTMPF,KCONF,KPRSF,KINTF,KWNDF,KFFTF,
        *KPUMP(2)
c--
c--     channel switching common block
c--
        COMMON/CHAN/IBCONB,IBCOO,IBFCI,IBVFI,IBVFO,IBTCM,IBREFR,
        *IBWS,IBMPR,IBPPR,IBPNIC,MBCON(7),NBCON,MBTMP(10),
        *NBTMP,MBPMP(2),NBPMP,MBPRS(7),NBPRS,MBFFT(2),NBFFT,
        *MBCONR(3),NBCONR
c--
c--     control character common block
c--
        COMMON /CONTRL/LA,LB,LC,LR,LDLE,LQU,LLAS,LTACT,LLACT,LTAC,
        *LLAC,LI,LF,LCR,LLF,ITRH1,ITRM1,ITRS1,ITRT1,ITRT2,ITRT3
c--
c--     diagnostic block
c--
        LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
        INTEGER*4 I1AREA,I2AREA
        COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
        *LTERM,ITERM,ATIMES,BTIMES,IADPOK,IADPEK,IPHONE,ILDTR,IHDTR,
        *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
        *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
        *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
        *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
        *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
        *JBOVR,JBWAIT,JBSLEP
c--
        DIMENSION ITMP(2)
c--
c---------------------------------------------------------------------
c--
c--     bg diagnostics
c--
        CALL IPOKEB (JBDS1,JBPMP)
c--
c--     timing failure traceback
c--
        CALL SCOPY ('  BOUPMP',LEVEL2)
        CALL SCOPY ('       ',LEVEL3)
c--
c--     teporary diagnostics for debugging
c--
D       CALL PUTSTR(LTERM,' LOCATION: ','0')
D       CALL PUTSTR(LTERM,LEVEL1,' ')
D       CALL PUTSTR(LTERM,LEVEL2,' ')
D       CALL PUTSTR(LTERM,LEVEL3,' ')
c--
c-----------------------------------
c--
c--     if pump status error flag set, skip this section
c--
        IF (KINTF.GT.0) GOTO 9
c--
c--     temp hold incase of handshake error
c--
        ITMP(1)=KFBR(1)
        ITMP(2)=KFBR(2)
c--
c--
        KCNT=LOCB       !number on stored in LOCB determined in BOUMH2
c--
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c--
c--     which pumps need to be turned on
c--
c--
c--     KCNT tells how many pumps need to be turned on
c--     if 0 then no pumps need to be on
c--     if 1 then branch to some pumps need on
c--
        IF (KCNT.NE.0) GOTO 52
c=========================================================================
c--
c--     pump decision section
c--
c=========================================================================
c--
c--
c--     no pumps need be on
c--
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c--
c--     what is current staus of plot and main pump
c--     1 on; then turn off
c--     0 off; leave off  branch to data storeage area
c--
21      IF (KFBP.EQ.0) GOTO 41
c--
c--     start sequence to turn pump off
c--     remember this is set up on a flip/flop chip which must be
```

```
c--     written to to turn the pump off
c--
20      CALL RLLTI(MFLAG)       !set scanner to close channels
        IF (MFLAG.EQ.1) GOTO 20
        CALL BOUCHN(IBMPR,MFLAG)        !main pump relay
        IF (MFLAG.EQ.1) GOTO 20
        CALL BOUCHN(IBPPR,MFLAG)        !plot pump relay
        IF (MFLAG.EQ.1) GOTO 20
c--
c--     store current staus of main and plot pumps
c--
c--
c--     determine which plot valve relays are on and turn off
c--
30      DO 40,L=1,NBPMP
c--
c--     determine if that plot valve relay is on
c--     if 0;off   - branch
c--     if 1;on  - turn off
c--
        IF (KFBR(L).EQ.0) GOTO 40
        CALL BOUCHN(MBPMP(L),MFLAG)     !plot valve relay
        IF (MFLAG.EQ.1) GOTO 20
        KFBR(L)=0       !set flag
        BOUDAT(29+L)=KFBR(L)    !store in array
40      CONTINUE
c--
c--     now close channels
c-
        CALL RET(MFLAG)
        IF (MFLAG.EQ.0) GOTO 41
        KFBR(1)=ITMP(1) !put back to status before handshake error
        KFBR(2)=ITMP(2)
        GOTO 20
c--
c--     reset pump flags to off (0)
c--
41      KFBP=0
c--
c--     store current staus
c--
        BOUDAT(26)=KFBP
        RETURN
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c--
c--     some pumps need be on
c--
c--
c--
c--     check to see if any change in status (for all relays) is needed
c--     if no --) return
c--     if yes --) change status
```

```
52      KP=0    !init flag for return
c--
        DO 53,I=1,NBPMP
        IF (KPUMP(I).LT.0 .AND. KFBR(I).EQ.0) GOTO 53  !off to stay off
        IF (KPUMP(I).GT.0 .AND. KFBR(I).EQ.1) GOTO 53  !on to stay on
        KP=KP+I
53      CONTINUE
        IF (KP.EQ.0) RETURN     !no change is needed
c--
c--     start sequence to turn pumps on
c--
50      CALL RLLTI(MFLAG)
        IF (MFLAG.EQ.1) GOTO 50
c--
c--     if pump already on skip this section
c--
51      IF (KFBP.EQ.1) GOTO 60
        CALL BOUCHN(IBMPR,MFLAG)        !main pump relay
        IF (MFLAG.EQ.1) GOTO 50
        CALL BOUCHN(IBPPR,MFLAG)        !plot pump relay
        IF (MFLAG.EQ.1) GOTO 50
c--
c--     do for the number of relays
c--
60      DO 440,KCP=1,NBPMP
c--
c--
c--     determine which plot pump valve relays are on
c--     and turn off
c--     if 0 off; skip
c--     if 1 on; turn off
c--
c--     determine what you should do in each state
c--
c--     KFBR    KPUMP   COMMENT
c--     ~~~~    ~~~~~   ~~~~~~~
c--     0       <0      relay needs to be off now off
c--     0       >0      relay needs to be on now off
c--     1       <0      relay now on needs to be off
c--     1       >0      relay now on needs to be on
c--
        IF (KFBR(KCP).EQ.0.AND.KPUMP(KCP).LT.0) GOTO 440        !leave be
        IF (KFBR(KCP).EQ.0.AND.KPUMP(KCP).GT.0) GOTO 65         !turn on
        IF (KFBR(KCP).EQ.1.AND.KPUMP(KCP).GT.0) GOTO 440        !leave be
        IF (KFBR(KCP).EQ.1.AND.KPUMP(KCP).LT.0) GOTO 65         !turn off
        TYPE *,'ERROR BOUPMP SHOULD NOT BE HERE'
        GOTO 440
c--
c--     write to flip/flop chip
c--
65      CALL BOUCHN(KPUMP(KCP),MFLAG)   !plot valve relay
        IF (MFLAG.EQ.1) GOTO 50
```

```
c--
c--     if kpump (0 not to be on if on turn off
c--              )0 to be on
c--
        IF (KPUMP(KCP).LT.0) GOTO 67
c--
c--     set pump flags to on by making them =1
c--     remember: KFBP flag for main and plot pumps is
c--     = to 1 if any plot valve on.
c--     get system date and time pump turned on
c--
        KFBR(KCP)=1
        BOUDAT(29+KCP)=KFBR(KCP)
        GOTO 440
67      KFBR(KCP)=0      !now off
c--
440     CONTINUE
c--
c--     done with plot valve relays?
c--     no - branch back and do again
c--     yes - OK now close the channels and store infor in proper array
c--
        CALL RET(MFLAG)
        IF (MFLAG.EQ.0) GOTO 75
        KFBR(1)=ITMP(1)
        KFBR(2)=ITMP(2)
        GOTO 50
c--
c--     reset pump flags to on (1)
c--
75      KFBP=1  !set main pump flag to on
c--
c--     store current staus
c--
        BOUDAT(26)=KFBP
        RETURN
c--
c--
c--     error ocurred
c--
9       WRITE (LTERM,*) 'BOULDER PUMP SECTION SKIPPED DUE TO
          KINTF FLAG SET'
        RETURN
        END
```

```
      SUBROUTINE BOUPRS
c +-----------------------------------------------------------------------+
c | Program :    BOULD.REL                                                |
c | Module  :    BOUPRS.FOR                                               |
c | Function:    READS THE HOMEMADE CONDUCTIVITY PROBES                   |
c |                                                                       |
c +-----------------------------------------------------------------------+
c--
c--     data common block
c--
        COMMON/BOULD/BOUDAT(60),BOUA(25),BOUB(25),BDATA(60),BOUT(60),
        *NARRAY,BCONAV(2),BTMPAV(2),BPRSAV(2),NPL,NPR,NBT1,NBT2,NBC1,
        *NBC2,NBP1,NBP2,LBCF(2),LBTF(2),LBPF(2),SETPTC(2),LOCB,
        *KFBR(2),KFBP,KSBR(2),KSBP,KTMPF,KCONF,KPRSF,KINTF,KWNDF,KFFTF,
        *KPUMP(2)
c--
c--     channel switching common block
c--
        COMMON/CHAN/IBCONB,IBCOO,IBFCI,IBVFI,IBVFO,IBTCM,IBREFR,
        *IBWS,IBMPR,IBPPR,IBPNIC,MBCON(7),NBCON,MBTMP(10),
        *NBTMP,MBPMP(2),NBPMP,MBPRS(7),NBPRS,MBFFT(2),NBFFT,
        *MBCONR(3),NBCONR
c--
c--     data checking common block
c--
        COMMON/CONDCL/WINDMX,TEMPMN,BCONL,BCONH,BCONRF(3),BRI(3),
        *BVOLTS,BAI,BTMPRF,BCONIN,BAVETP(2),BTMPHZ,RANL,RANH,BFTDAC(4)
c--
c--     control character common block
c--
        COMMON /CONTRL/LA,LB,LC,LR,LDLE,LQU,LLAS,LTACT,LLACT,LTAC,
        *LLAC,LI,LF,LCR,LLF,ITRH1,ITRM1,ITRS1,ITRT1,ITRT2,ITRT3
c--
c--     diagnostic block
c--
        LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
        INTEGER*4 I1AREA,I2AREA
        COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
        *LTERM,ITERM,ATIMES,BTIMES,IADPOK,IADPEK,IPHONE,ILDTR,IHDTR,
        *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
        *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
        *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
        *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
        *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
        *JBOVR,JBWAIT,JBSLEP
c--
c--     ------------------------------------------------------------------
c--
c--
c--     bg program diagnostics
c--
        CALL IPOKEB(JBDS1,JBPRS)
```

```
c--
c--     timing failure traceback
c--
        CALL SCOPY (' BOUPRS',LEVEL2)
        CALL SCOPY ('       ',LEVEL3)
c--
c--     temporary diagnostics for debugging
c--
D       CALL PUTSTR(LTERM,' LOCATION: ','0')
D       CALL PUTSTR(LTERM,LEVEL1,' ')
D       CALL PUTSTR(LTERM,LEVEL2,' ')
D       CALL PUTSTR(LTERM,LEVEL3,' ')
c--
c-----------------------------------------------------------------------
c--
c--             channel switching section stats here
c--             see boudat.for for channel assignments
c--
c-----------------------------------------------------------------------
c--
        DO 30,L=1,NBPRS
c--
c--     determine which ave temp to use
c--
        IF (L .LE.1) BTMP=BOUDAT(1)      !reference temperature
        IF (L .GE.2.AND. L .LE.4) BTMP=BTMPAV(1)        !ave for plot 1
        IF (L .GE.5) BTMP=BTMPAV(2)      !ave for plot 2
c--
c--
c--     locb location is equal to
c--     number of temperatures +number of conductivities
c--     +1 for the oscillator
c--
        LOCB=(L+1)+NBTMP+NBCON  !array locator for boudat array
        LTEST=1 !conductivity range reference location
c--
20      CALL RLLTI(MFLAG)        !sets conditions so channels can be closed
        IF (MFLAG.EQ.1) GO TO 20
c--
c--     close voltage to frequency convertor ouput and test error flag
c--
        CALL BOUCHN (IBVFO,MFLAG)
        IF (MFLAG.EQ.1) GO TO 20
c--
c--     close frequency counter input
c--
        CALL BOUCHN (IBFCI,MFLAG)
        IF (MFLAG.EQ.1) GO TO 20
c--
c--     close voltage to frequency input
c--
        CALL BOUCHN (IBVFI,MFLAG)
```

```
      IF (MFLAG.EQ.1) GO TO 20
c--
c--   close conductance bridge
c--
      CALL BOUCHN (IBCONB,MFLAG)
      IF (MFLAG.EQ.1) GO TO 20
c--
c--   close specified channel
c--
      CALL BOUCHN (MBPRS(L),MFLAG)
      IF (MFLAG.EQ.1) GO TO 20
c--
c--   close range selector
c--
      CALL BOUCHN (MBCONR(LTEST),MFLAG)
      IF (MFLAG.EQ.1) GO TO 20
c--
      CALL RET(MFLAG) !tells scanner all commands have been sent
      IF (MFLAG.EQ.1) GO TO 20
c--
      CALL RTLLI(MFLAG)        !sets conditions so data can be received
      IF (MFLAG.EQ.1) GO TO 20
c--
      CALL WRITE(BOUDAT(LOCB),MFLAG)  !converts data to real*4 number
      IF (MFLAG.EQ.1) GO TO 20
c--
c--
      LTEST=LTEST+1   !increment reference location channel
      IF (LTEST.GT.3) GO TO 40         !error only three ranges
c--
c--   is reading within proper range
c--
      IF (BOUDAT(LOCB).GT.BCONH.OR.BOUDAT(LOCB).LT.BCONL) GO TO 20
      LTEST=LTEST-1   !decrement to keep proper range after test
c--
c--
c--   check conductance board if first time thru
c--    check to see if in 20% tolerance range (1.2 (-) 0.8)
c--
      IF (L.NE.1) GO TO 333
      IF ((BOUDAT(LOCB).GT.(1.2*BCONRF(LTEST))).OR.(3OUDAT(LOCB).LT.
       (0.8*BCONRF(LTEST)))) GOTO 60
c--
c--
c--   do conductivity calculations
c--    1 volt = 10 Khz
c--     divide by 10 to get reading back into volts
c--   conductivity= comductivity+ teperature correction
c--     temp correction =[(Ref.temp.) - ((ave temp)*cond.correction)*10]
c--           *1000 to convert to micro/Siemens
c--
333   TMP=BOUDAT(LOCB)/(10.*BRI(LTEST)*BVOLTS)
```

```
        BOUDAT(LOCB)=TMP*(1+(BTMPRF-BTMP)*BCONIN)*10.E+06
c--
c--
D       WRITE (LTERM,*) WA,WB,BOUDAT(LOCB),LOCB,LTEST,BCONRF(LTEST)
        GOTO 30
40      WRITE (LTERM,41) L
41      FORMAT (////' PROBLEM IN CONDUCTIVITY RANGE. CHANNEL ',I5,////)
30      CONTINUE
        RETURN
c--
c-------------------------------------------
c--
c--     section to test where conductance problem is
c--
c-------------------------------------------
c--
c--     error conductance board failure
c--
60      KPRSF=1
        IPOK=(IPEEKB(JBDERR))+JBETMP
        CALL IPOKEB(JBDERR,IPOK)
        RETURN
        END
```

```
      SUBROUTINE BOUSLP
c +------------------------------------------------------------------+
c | Program :    BOULD.REL                                           |
c | Module  :    BOUSLP.FOR                                          |
c | Function:    DETERMINES WAIT PERIOD BETWENN DATA SAMPLING        |
c |                                                                  |
c +------------------------------------------------------------------+
c--
c--   diagnostic block
c--
      LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
      INTEGER*4 I1AREA,I2AREA
      COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
      *LTERM, ITERM, ATIMES, BTIMES, IADPOK, IADPEK, IPHONE, ILDTR, IHDTR,
      *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
      *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
      *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
      *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
      *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
      *JBOVR,JBWAIT,JBSLEP
c--
c--   file information block
c--
      COMMON/FILE/BOUDEV,MABLK(4),MBBLK(4),MCBLK(4),MDBLK(4),
      *MEBLK(4),MFBLK(4),MGBLK(4),MHBLK(4),MIBLK(4),MJBLK(4),MKBLK(4),
      *MLBLK(4),MMBLK(4),MNBLK(4),MOBLK(4),MPBLK(4),MQBLK(4),MRBLK(4),
      *MSBLK(4),
      *MABLKL,MABLKW,MBBLKL,MBBLKW,MCBLKL,
      *MCBLKW,MDBLKL,MDBLKW,MEBLKL,MEBLKW,MFBLKL,MFBLKW,MGBLKL,
      *MGBLKW,MHBLKL,MHBLKW,MIBLKL,MIBLKW,MJBLKL,MJBLKW,MKBLKL,
      *MKBLKW,MLBLKL,MLBLKW,MMBLKL,MMBLKW,MNBLKL,MNBLKW,MOBLKL,
      *MOBLKW,MPBLKL,MPBLKW,MQBLKL,MQBLKW,MRBLKL,MRBLKW,MSBLKL,MSBLKW
c--
c--   main program sleep block
c--
      COMMON/WAIT/IBPOLH,IBPOLM,IBPOLS,IBPOLT,IBCC,IBCH,IBCM,
      *IBCS,IBCT
c--
c--
c--
      DIMENSION DAYS(12),ZWRITE(2)
      DATA DAYS/31.,28.,31.,30.,31.,30.,31.,31.,30.,
      *31.,30.,31./
c--
c-------------------------------------------------------------------
c--   array locations for time and date of next run
c--   these values are calculated from the system date and time
c--   ZWIRTE(1)=time
c--   ZWRITE(2)=date
c-------------------------------------------------------------------
c--
c--   bg diagnostics
```

```
c--
        CALL IPOKEB(JBDS1,JBSLP)
c--
c--     timing failure traceback
c--
        CALL SCOPY ('   BOUSLP',LEVEL3)
c--
c--     temporary diagnostics for debugging
c--
D       CALL PUTSTR(LTERM,' LOCATION: ','0')
D       CALL PUTSTR(LTERM,LEVEL1,' ')
D       CALL PUTSTR(LTERM,LEVEL2,' ')
D       CALL PUTSTR(LTERM,LEVEL3,' ')
c--
c--     calc time for sleep in real *4
c--     use current time in subroutine and not systim as this program maye
c--      5-8 minutes to run and if you are using short waiting periods
c--      you will get the wrong time in the time data file.
c--
        CURTIM=SYSTIM(DTMP)
        TMP=(FLOAT(IBPOLH)*100.)+FLOAT(IBPOLM)
        TIME=TIMADD(TMP,CURTIM,IFLAG)
c--
c--     is this a new day ? IFLAG=1
c--
        DATE=SYSDAT
        IF (IFLAG.EQ.0) GOTO 12
        TIME=TIME-2400.
        CALL BOUDYR(SYSDAT,DATE)
c--
c--
c--
12      ZWRITE(1)=TIME
        ZWRITE(2)=DATE
c--
c--     write to disk file
c--
        CALL BOURED (MJBLK,ZWRITE,MJBLKW/2,MJBLKW,0,'WR','BO',LERR)
        IF (LERR.NE.0) GOTO 20
        IBCH=IBPOLH
        IBCM=IBPOLM
        IBCS=IBPOLS
        IBCT=IBPOLT
        RETURN
c--
c--     error on write
c--
20      WRITE (LTERM,*) 'SLP-EW MJBLK'
        RETURN
        END
```

```
      SUBROUTINE BOUSTA
c +------------------------------------------------------------------+
c | Program :    BOULD.REL                                            |
c | Module  :    BOUSTA.FOR                                           |
c | Function:    OUTPUTS STATUS INFORMATION TO DESIGNATED DEVICE      |
c |                                                                   |
c +------------------------------------------------------------------+
c--
c--      data common block
c--
         COMMON/BOULD/BOUDAT(60),BOUA(25),BOUB(25),BDATA(60),BOUT(60),
         *NARRAY,BCONAV(2),BTMPAV(2),BPRSAV(2),NPL,NPR,NBT1,NBT2,NBC1,
         *NBC2,NBP1,NBP2,LBCF(2),LBTF(2),LBPF(2),SETPTC(2),LOCB,
         *KFBR(2),KFBP,KSBR(2),KSBP,KTMPF,KCONF,KPRSF,KINTF,KWNDF,KFFTF,
         *KPUMP(2)
c--
c--      data checking common block
c--
         COMMON/CONDCL/WINDMX,TEMPMN,BCONL,BCONH,BCONRF(3),BRI(3),
         *BVOLTS,BAI,BTMPRF,BCONIN,BAVETP(2),BTMPHZ,RANL,RANH,BFTDAC(4)
c--
c--      diagnostic block
c--
         LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
         INTEGER*4 I1AREA,I2AREA
         COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
         *LTERM,ITERM,ATIMES,BTIMES,IADPOK,IADPEK,IPHONE,ILDTR,IHDTR,
         *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
         *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
         *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
         *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
         *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
         *JBOVR,JBWAIT,JBSLEP
c--
c--      <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <> <>
c--
c--      dimension labels and arrays
c--

         DIMENSION XLABEL(3),YLABEL(3),ZLABEL(3),
         *QLABEL(3),RLABEL(3),ULABEL(3),VLABEL(3),
         *STAT(2),MLOC(2,11),ALABEL(3),BLABEL(3)
         LOGICAL*1 TLABEL(8),DLABEL(9)
c--
c--      data for labels
c--
         DATA XLABEL/'TEMP','ERAT','URE '/YLABEL/
         *'COND','UCTI','VITY'/ZLABEL/'COMM','. CO',
         *'ND  '/ZON/'ON  '/ZOFF/'OFF '/QLABEL/'PLOT',
         *' STA','TUS '/RLABEL/'SET ','POINT','T   '/
         *DEP1/'30cm'/ULABEL/'WIND','SPEE','D   '/
         *WLABEL/'AIR '/VLABEL/'FLOW',' RAT','E   '/
```

```
        *ALABEL/'COND','. RE','AD. '/BLABEL/'DIFF','EREN',
        *'CE  '/
c--
c--     MLOC array is used to point to the proper locqation of the BOUDAT
c--     array to allow the data output to use a do loop for ease of formag
c--     see boudat.dat for locations of boudat array.
c--
        DATA MLOC/2,6,3,7,4,8,12,15,13,16,14,17,19,22,20,23,21,24,
        *5,9,28,29/
c--
c--     temp output to detect errors
c--
c--
c--     bg diagnostics
c--
        CALL IPOKEB (JBDS1,JBSTA)
c--
c--     timing failure traceback
c--
        CALL SCOPY (' BOUSTA',LEVEL2)
c--
c--     get date and time to be output with data
c--
        CALL TIME (TLABEL)
        CALL DATE(DLABEL)
c--
c--     write header
c--
        WRITE (LTERM,9)          !FORMFEED
        WRITE (LTERM,10) BTIMES
c--
c--     initialize pump staus flag to "off"
c--     then determine if pumps were on
c--
        DO 101,I=1,NPL
        STAT(I)=ZOFF
101     IF (BOUDAT(29+I).EQ.1.) STAT(I)=ZON      !LOC 30,31
c--
c--     output wind speed and air temperature
c--
        WRITE (LTERM,40) DLABEL,TLABEL
        WRITE (LTERM,29) ULABEL,BOUDAT(25)       !WIND SPEED
        WRITE (LTERM,28) WLABEL,XLABEL,BOUDAT(10)       !AIR TEMP
        WRITE (LTERM,11)
        DO 100,L=1,2
100     WRITE (LTERM,12) L
        WRITE (LTERM,15)
c--
c--     set pump status and output ave temp, cond, comm. cond
c--     along with the setpoint
c--
        WRITE (LTERM,16) XLABEL,BOUDAT(45),BOUDAT(46)
        WRITE (LTERM,16) ALABEL,BOUDAT(RANL),BOUDAT(RANH)
        WRITE (LTERM,16) RLABEL,SETPTC(1),SETPTC(2)
        WRITE (LTERM,16) BLABEL,BOUDAT(RANL)-SETPTC(1),
        *BOUDAT(RANH)-SETPTC(2)
        WRITE (LTERM,17) QLABEL,STAT(1),STAT(2)
c--
c--     output data for both plots
c--
        DO 25,I=1,NPL
        LM=0    !INITIALIZE ARRAY SUBSCRIPT FOR OUTPUT
        WRITE (LTERM,20) I
        DO 26 L=1,NPR
        WRITE (LTERM,21) L
26      CONTINUE
        WRITE (LTERM,15)
        WRITE (LTERM,27) XLABEL,BOUDAT(MLOC(I,LM+1)),BOUDAT(MLOC(I,LM+2)),
        *BOUDAT(MLOC(I,LM+3))   !LOC (2-4) (6-8)
        WRITE (LTERM,27) YLABEL,BOUDAT(MLOC(I,LM+4)),BOUDAT(MLOC(I,LM+5)),
        *BOUDAT(MLOC(I,LM+6))   !LOC (12-14) (15-17)
        WRITE (LTERM,27) ZLABEL,BOUDAT(MLOC(I,LM+7)),BOUDAT(MLOC(I,LM+8)),
        *BOUDAT(MLOC(I,LM+9))   !LOC (19-21) (22-24)
        WRITE (LTERM,28) XLABEL,DEP1,BOUDAT(MLOC(I,LM+10))      !LOC (5,9)
        WRITE (LTERM,29) VLABEL,BOUDAT(MLOC(I,LM+11))   !LOC (28,29)
25      CONTINUE
        WRITE (LTERM,9)         !FORMFEED
        RETURN
c--
c--------------------------------------------------------------------------
c--
c--     format statements
c--------------------------------------------------------------------------
9       FORMAT (1H1)
10      FORMAT (T30,' Boulder Status Information',F10.0/)
11      FORMAT (//,' Average Readings',$)
12      FORMAT (T20,'Plot: ',I1,$)
15      FORMAT (/,' ')
16      FORMAT (X,3A4,T33,F10.2,T56,F10.2)
17      FORMAT (X,3A4,T38,A4,T61,A4)
20      FORMAT (//,' Plot ',I1,' Readings',$)
21      FORMAT (T13,'Location: ',I1,$)
27      FORMAT (X,3A4,T27,F10.2,T48,F10.2,T69,F10.2)
28      FORMAT (X,4A4,T27,F10.2)
29      FORMAT (X,3A4,T27,F10.2)
40      FORMAT (' Date: ',9A1,/,' Time: ',8A1,//)
        END
```

```
      SUBROUTINE BOUTIM
c +-----------------------------------------------------------------+
c | Program :    BOULD.REL                                          |
c | Module  :    BOUTIM.FOR                                         |
c | Function:    CALCULATES THE AMOUNT TIME EACH PUMP WAS ON        |
c |                                                                 |
c +-----------------------------------------------------------------+
c--
c--     data common block
c--
        COMMON/BOULD/BOUDAT(60),BOUA(25),BOUB(25),BDATA(60),BOUT(60),
        *NARRAY,BCONAV(2),BTMPAV(2),BPRSAV(2),NPL,NPR,NBT1,NBT2,NBC1,
        *NBC2,NBP1,NBP2,LBCF(2),LBTF(2),LBPF(2),SETPTC(2),LOCB,
        *KFBR(2),KFBP,KSBR(2),KSBP,KTMPF,KCONF,KPRSF,KINTF,KWNDF,KFFTF,
        *KPUMP(2)
c--
c--     diagnostic block
c--
        LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
        INTEGER*4 I1AREA,I2AREA
        COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
        *LTERM,ITERM,ATIMES,BTIMES,IADPOK,IADPEK,IPHONE,ILDTR,IHDTR,
        *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
        *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
        *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
        *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
        *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
        *JBOVR,JBWAIT,JBSLEP
c--
c----------------------------------------------------------------------
c--     subroutine calls:
c--      DATEI - converts a REAL*4 date into 3 REAL*4 variables
c--        which are year,month,day
c--      TDATE - returns the difference in days between two dates
c--
c----------------------------------------------------------------------
c--
        DIMENSION XTIME(2)       !2 FOR NUMBER PLOTS
c--
c--     bg diagnostics
c--
        CALL IPOKEB(JBDS1,JBTIM)
c--
c--     timinf failure traceback
c--
        CALL SCOPY ('  BOUTIM',LEVEL2)
        CALL SCOPY ('        ',LEVEL3)
c--
c--     temporary diagnostics for debugging
c--
D       CALL PUTSTR(LTERM,' LOCATION: ','0')
D       CALL PUTSTR(LTERM,LEVEL1,' ')
```

```
D       CALL PUTSTR(LTERM,LEVEL2,' ')
D       CALL PUTSTR(LTERM,LEVEL3,' ')
c--
c--     determine if there was an initialization failure
c--     if so skip to return and note this failure
c--
        IF (KINTF.LE.0) GOTO 9
        WRITE (LTERM,*) 'Boulder time section skipped
        *due to kintf flag set'
        RETURN
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c--     calculate time pumps were on
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c--
c--
c--     init back to zero
c--
9       XHRS=0.
        DO 2,I=1,NPL
2       XTIME(I)=0.
D       WRITE (LTERM,*) 'BOUTIM',KSBP,KSBR(1),KSBR(2),
D       *KFBP,KFBR(1),KFBR(2)
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c--
c--     if main pump was off no need to determine time on
c--
c--     past and present off
c--
        IF (KSBP.EQ.0.AND.KFBP.EQ.0) RETURN
c--
c--     past on   present off
c--
        IF (KSBP.EQ.1.AND.KFBP.EQ.0) GOTO 10
c--
c--     past and present on
c--
        IF (KSBP.EQ.1.AND.KFBP.EQ.1) GOTO 10
c--
c--     past off   present on
c--
        IF (KSBP.EQ.0.AND.KFBP.EQ.1) GOTO 30
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c--
c--     determine time pumps were on
c--
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
10      DO 11,L=1,NPL
D       WRITE (LTERM,*) 'PUMPS ON',L,KFBR(L),KSBR(L)
c--
c--     was plot relay on the time before?
c--
        IF (KFBR(L).EQ.0) GOTO 113
```

```
c--
c--     determine time pump was on
c--
        CALL DATEI (BOUDAT(33+L),A,B,C)
        CALL DATEI (SYSDAT,D,E,F)
        CALL TDATE(IFIX(A),IFIX(B),IFIX(C),IFIX(D),IFIX(E),IFIX(F),IDAY)
        XHRS=IDAY*24
        XTIME(L)=TIMDIF(SYSTM1,BOUDAT(31+L))
        XTIME(L)=XHRS+XTIME(L)
D       WRITE (LTERM,*) 'XTIME',L,XTIME(L)
c--
c--     is relay on this time ?
c--
113     IF (KFBR(L).EQ.1) GOTO 112
        BOUDAT(50+L)=XTIME(L)
        BOUDAT(54+L)=SYSTM1
        BOUDAT(56+L)=SYSDAT
        GOTO 11
c--
c--     reset time on
c--
112     BOUDAT(31+L)=0
        BOUDAT(33+L)=0
11      CONTINUE
        RETURN
c--
c--     set new system date and time for calculations
c--
30      DO 31,L=1,NPL
D       WRITE (LTERM,*) 'PAST ON PRESENT OFF'
c--
c--     if pump now off branch
c--
        IF (KFBR(L).EQ.0) GOTO 31
c--
c--     pump on set new date and time
c--
        BOUDAT(31+L)=SYSTM1
        BOUDAT(33+L)=SYSDAT
31      CONTINUE
        RETURN
        END
```

```
      SUBROUTINE BOUTMP
c +---------------------------------------------------------------------+
c | Program :    BOULD.REL                                              |
c | Module  :    BOUTMP.FOR                                             |
c | Function:    READS TEMPERATURE MODULE                               |
c |                                                                     |
c +---------------------------------------------------------------------+
c +---------------------------------------------------------------------+
c |   NOTE:                                                             |
c |                                                                     |
c |        ktmpf=1 reference temp out of range                          |
c |        ktmpf=2 module failure?                                      |
c |        ktmpf=3 module lockup                                        |
c +---------------------------------------------------------------------+
c--
c--     data common block
c--
      COMMON/BOULD/BOUDAT(60),BOUA(25),BOUB(25),BDATA(60),BOUT(60),
     *NARRAY,BCONAV(2),BTMPAV(2),BPRSAV(2),NPL,NPR,NBT1,NBT2,NBC1,
     *NBC2,NBP1,NBP2,LBCF(2),LBTF(2),LBPF(2),SETPTC(2),LOCB,
     *KFBR(2),KFBP,KSBR(2),KSBP,KTMPF,KCONF,KPRSF,KINTF,KWNDF,KFFTF,
     *KPUMP(2)
c--
c--     channel switching common block
c--
      COMMON/CHAN/IBCONB,IBCOO,IBFCI,IBVFI,IBVFO,IBTCM,IBREFR,
     *IBWS,IBMPR,IBPPR,IBPNIC,MBCON(7),NBCON,MBTMP(10),
     *NBTMP,MBPMP(2),NBPMP,MBPRS(7),NBPRS,MBFFT(2),NBFFT,
     *MBCONR(3),NBCONR
c--
c--     data checking common block
c--
      COMMON/CONDCL/WINDMX,TEMPMN,BCONL,BCONH,BCONRF(3),BRI(3),
     *BVOLTS,BAI,BTMPRF,BCONIN,BAVETP(2),BTMPHZ,RANL,RANH,BFTDAC(4)
c--
c--     control character common block
c--
      COMMON /CONTRL/LA,LB,LC,LR,LDLE,LQU,LLAS,LTACT,LLACT,LTAC,
     *LLAC,LI,LF,LCR,LLF,ITRH1,ITRM1,ITRS1,ITRT1,ITRT2,ITRT3
c--
c--     diagnostic block
c--
      LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
      INTEGER*4 I1AREA,I2AREA
      COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
     *LTERM,ITERM,ATIMES,BTIMES,IADPOK,IADPEK,IPHONE,ILDTR,IHDTR,
     *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
     *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
     *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
     *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
     *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
     *JBOVR,JBWAIT,JBSLEP
```

```
c--
c-----------------------------------------------------------------------
c--
c--
c--
c--     ITMPF is an error flag which is set when there is an error with te
c--     temperature module. initialize to zero each time called
c--
        KTMPF=0 !init flag
c--
c--     diagnostics section
c--
        CALL IPOKEB(JBDS1,JBTMP)
        CALL SCOPY (' BOUTMP',LEVEL2)
        CALL SCOPY ('       ',LEVEL3)
D       CALL PUTSTR(LTERM,' LOCATION: ','0')
D       CALL PUTSTR(LTERM,LEVEL1,' ')
D       CALL PUTSTR(LTERM,LEVEL2,' ')
D       CALL PUTSTR(LTERM,LEVEL3,' ')
c--
c--     read temperature and check for equipment failure
c--
        ILOCK=0          !for lockup check
        IZERO=0          !for module check
c--
        DO 30,L=1,NBTMP
        ITRY=0  !init zero read counter
        LOCB=L  !array location for boudat
c-----------------------------------------------------------------------
c--     start control sequence
c--
20      CALL RLLTI(MFLAG)        !set conditions so channels can be closed
        IF (MFLAG.EQ.1) GO TO 20
c--
c--     close voltage to frequency counter input
c--
        CALL BOUCHN (IBFCI,MFLAG)
        IF (MFLAG.EQ.1) GO TO 20
c--
c--
c--     close voltage frequency output
c--
        CALL BOUCHN (IBVFO,MFLAG)
        IF (MFLAG.EQ.1) GO TO 20
c--
c--     close voltage frequency input
c--
        CALL BOUCHN (IBVFI,MFLAG)
        IF (MFLAG.EQ.1) GO TO 20
c--
c--
c--     close temperature conversion module
```

```
c--
        CALL BOUCHN (IBTCM,MFLAG)
        IF (MFLAG.EQ.1) GO TO 20
c--
c--     close temperature sensor for plot
c--
        CALL BOUCHN (MBTMP(L),MFLAG)
        IF (MFLAG.EQ.1) GO TO 20
c--
        CALL RET(MFLAG) !tells scanner all commands have been sent
        IF (MFLAG.EQ.1) GO TO 20
c--
        CALL ISLEEP (0,0,1,0)   !give time for sensors to =
c--
        CALL RTLLI(MFLAG)       !sets conditions so data can be received
        IF (MFLAG.EQ.1) GO TO 20
c--
        CALL WRITE(BOUDAT(LOCB),MFLAG)  !converts data to real*4 number
        IF (MFLAG.EQ.1) GO TO 20
D       WRITE (LTERM,*) 'BOUTMP ',BOUDAT(LOCB),LOCB
c--
c--     end control sequence
c-------------------------------------------------------------------------
c--
c--     if conductivities and other temperature are 0 also
c--     scanner is malfunctioning
c--     calculate temperatures for all columns
c--
c--     check for zero condition
c--     a zero may not be a zero but a bad reading ckeck and see
c--
        ITRY=ITRY+1
        IF (ITRY.LE.5
           .AND.BOUDAT(LOCB) .EQ. 0.)
               GOTO 20
        IF (BOUDAT(LOCB).EQ.0.
           .AND. ITRY .GT.5)
               WRITE (LTERM, 222) LOCB,BOUDAT(LOCB)
        IF (BOUDAT(LOCB).LT.100.) GOTO 200
        ILOCK=ILOCK+1
        WRITE (LTERM,*) 'BOUTMP - lockup probe ',LOCB
200     IF (BOUDAT(LOCB).NE.0.) GOTO 334
c--
c--     set zero counter for module check
c--
        IZERO=IZERO+1
c--
c--     do calculations
c--     Equation:
c--
c--     degree c = ([data/10.] + constant)/0.1
c--
```

```
c--             where data is in Khz
c--
c--      BOUA array holds temperature constant
c--      0.1  -) 100 millivolts per degree centigrade
c--      10. -) 10 Khz = 1volt
c--
334      BOUDAT(LOCB)=((BOUDAT(LOCB)/10.)+BOUA(LOCB))/BOUB(LOCB)
c--
c--      check reference temp resistor
c--
D        WRITE (LTERM,9001) BOUDAT(LOCB),BOUA(LOCB),BOUB(LOCB),LOCB
D9001     FORMAT (X,5X,'BOUTMP',E12.5,5X,' BOUA',E12.5,
D        *'  BOUB',E12.5,'  LOC',I5,10A1)
30       CONTINUE
c------------------------------------
c--      error section
c--      temp module failure ?????
c--      possible scanner failure
c--------------------------------------
D        WRITE (LTERM,*) 'BOUTMP',BOUDAT(1),(2.0*BTMPHZ),(0.0*BTMPHZ)
         IF ((BOUDAT(1) .LE. (0.0*BTMPHZ))
             .OR. (BOUDAT(1) .GT. (2.0*BTMPHZ)))
                  GOTO 60
         RETURN
c--
c--      set error flag to indicate temperature module check has taken plae
c--
60       KTMPF=1          !deviation error
D        WRITE (LTERM,*) 'TEMP ERROR'
c--
c--      is it module failure or deviation of equipment??
c--      IZERO is a counter to determine how many probes are = to 0
c--      if all probes = 0 then there is a module failure.
c--
         IF (IZERO.EQ.NBTMP) KTMPF=2     !board failure low end
c--
c--      lockup of module
c--
         IF (ILOCK.GT.2) KTMPF=3          !lockup error
         IF (BOUDAT(1) .GE. 50.) KTMPF=4 !lockup error
c--
c--      failure
c--
D        WRITE (LTERM,*) 'BOUTMP KTMPF',KTMPF,'ILOCK ',ILOCK,
D        *'IZERO ',IZERO
400      RETURN
222      FORMAT (' TEMPERATURE ERROR ARRAY LOCATION',I5,3X
          'READING',E12.5)
         END
```

```
      SUBROUTINE BOUWND
c +-----------------------------------------------------------------------+
c | Program :    BOULD.REL                                                 |
c | Module  :    BOUWND.FOR                                                |
c | Function:    READS WIND SPEED MODULE                                   |
c |                                                                        |
c +-----------------------------------------------------------------------+
c--
c--
c--   data common block
c--
      COMMON/BOULD/BOUDAT(60),BOUA(25),BOUB(25),BDATA(60),BOUT(60),
     *NARRAY,BCONAV(2),BTMPAV(2),BPRSAV(2),NPL,NPR,NBT1,NBT2,NBC1,
     *NBC2,NBP1,NBP2,LBCF(2),LBTF(2),LBPF(2),SETPTC(2),LOCB,
     *KFBR(2),KFBP,KSBR(2),KSBP,KTMPF,KCONF,KPRSF,KINTF,KWNDF,KFFTF,
     *KPUMP(2)
c--
c--   channel switching common block
c--
      COMMON/CHAN/IBCONB,IBCOO, IBFCI, IBVFI, IBVFO, IBTCM, IBREFR,
     *IBWS, IBMPR, IBPPR, IBPNIC,MBCON(7),NBCON, MBTMP(10),
     *NBTMP, MBPMP(2),NBPMP,MBPRS(7),NBPRS, MBFFT(2),NBFFT,
     *MBCONR(3),NBCONR
c--
c--   data checking common block
c--
      COMMON/CONDCL/WINDMX,TEMPMN,BCONL,BCONH,BCONRF(3),BRI(3),
     *BVOLTS,BAI,BTMPRF,BCONIN,BAVETP(2),BTMPHZ,RANL,RANH,BFTDAC(4)
c--
c--   control character common block
c--
      COMMON /CONTRL/LA,LB,LC,LR,LDLE,LQU,LLAS,LTACT,LLACT,LTAC,
     *LLAC,LI,LF,LCR,LLF,ITRH1,ITRM1,ITRS1,ITRT1,ITRT2,ITRT3
c--
c--   diagnostic block
c--
      LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
      INTEGER*4 I1AREA,I2AREA
      COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
     *LTERM,ITERM,ATIMES,BTIMES,IADPOK,IADPEK,IPHONE,ILDTR,IHDTR,
     *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
     *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
     *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
     *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
     *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
     *JBOVR,JBWAIT,JBSLEP
c-------------------------------------------------------------------------
c--
c--   dianostic section
c--
      KWNDF=0 !init flag
c--
```

```
c--     bg program diagnostics
c--
        CALL IPOKEB(JBDS1,JBWND)
c--
c--     timing failure traceback
c--
        CALL SCOPY (' BOUWND',LEVEL2)
        CALL SCOPY ('       ',LEVEL3)
c--
c--     tempory diagnostics for debugging
c--
D       CALL PUTSTR(LTERM,' LOCATION: ','0')
D       CALL PUTSTR(LTERM,LEVEL1,' ')
D       CALL PUTSTR(LTERM,LEVEL2,' ')
D       CALL PUTSTR(LTERM,LEVEL3,' ')
c--
c--------------------------------------------------------------------------
c--
c--             channel switching section starts here
c--             see BOUDAT.FOR for channel assignments
c--------------------------------------------------------------------------
        ITRY=0  !init zero check
        LOCB=25         !array locator for boudat array
c--
c--
c--
20      CALL RLLTI(MFLAG)       !sets conditions so channels can be closed
        IF (MFLAG.EQ.1) GO TO 20
c--
c--
c--     close voltage to frequency convertor ouput and test error flag
c--
        CALL BOUCHN (IBVFO,MFLAG)
        IF (MFLAG.EQ.1) GO TO 20
c--
c--     close frequency counter input
c--
        CALL BOUCHN (IBFCI,MFLAG)
        IF (MFLAG.EQ.1) GO TO 20
c--
c--     close voltage to frequency input
c--
        CALL BOUCHN (IBVFI,MFLAG)
        IF (MFLAG.EQ.1) GO TO 20
c--
c--     close wind speed channel
c--
        CALL BOUCHN (IBWS,MFLAG)
        IF (MFLAG.EQ.1) GO TO 20
c--
        CALL RET(MFLAG) !tells scanner all commands have been sent
        IF (MFLAG.EQ.1) GO TO 20
```

```
c--
        CALL RTLLI(MFLAG)        !sets conditions so data can be received
        IF (MFLAG.EQ.1) GO TO 20
c--
        CALL WRITE(BOUDAT(LOCB),MFLAG)  !converts data to real*4 number
        IF (MFLAG.EQ.1) GO TO 20
c--
c--     check for zero condition
c--     a zero may not be a zero but a bad reading ckeck and see
c--
        ITRY=ITRY+1
        IF (ITRY.LE.5 .AND.BOUDAT(LOCB) .EQ. 0.) GOTO 20
c--
c--     do calculations for wind speed
c--
c--     calculations:
c--      [1kHz=10mph]  reading is in mega Hz so it is converted to kHz by
c--      divideby 100. which yields mph .This is converted to Km/hr
c--      by multiplying by 1.609
c--
c--     divide by 100 because 10kHz = 100mph or 1kHz=10mph
c--     1.609 to convert to Km/hr
c--
333     BOUDAT(LOCB)=(BOUDAT(LOCB)/100.)*1.609  !result in Km/hr
c--
c--
D       WRITE (LTERM,9000) BOUDAT(LOCB),LOCB,LEVEL2
D9000   FORMAT (X,5X,'READING',E12.5,5X,'LOC',I5,10A1)
30      CONTINUE
        RETURN
        END
```

```
      SUBROUTINE BOUBIT
c +-----------------------------------------------------------------------+
c | Program :    BOULD.REL                                                |
c | Module  :    BOULBIT.FOR                                              |
c | Function:    converts integer to binary bit equivalent for determinin|
c |              status of DAC and system configuration                   |
c |                                                                       |
c +-----------------------------------------------------------------------+
c |    in array ibit(16)    ibit(1)=bit 0  ibit(16)=bit 15                |
c +-----------------------------------------------------------------------+
c--
c--   diagnostic block
c--
      LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
      INTEGER*4 I1AREA,I2AREA
      COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
      *LTERM,ITERM,ATIMES,BTIMES,IADPOK,IADPEK,IPHONE,ILDTR,IHDTR,
      *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
      *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
      *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
      *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
      *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
      *JBOVR,JBWAIT,JBSLEP
c--
c--   base address common block
c--
      COMMON/PROCES/IAD103,IAD123,ITM03,ITM23
c--
c--   bits common
c--
      COMMON/BITS/IBIT(16),IWORD
c--
      DIMENSION KBIT(16)
c--
c--
c-----------------------------------------------------------------------
c--
c--   set diagnostic byte fo bg program
c--
      CALL IPOKEB (JBDS2,JBBIT)
c--
c--   temporary diagnostics for debugging
c--
D     CALL PUTSTR(LTERM,' LOCATION: ','0')
D     CALL PUTSTR(LTERM,LEVEL1,' ')
D     CALL PUTSTR(LTERM,LEVEL2,' ')
D     CALL PUTSTR(LTERM,LEVEL3,' ')
c--
c--    initialize array to zeros
c--
      DO 5 INDEX = 1,16
    5   KBIT(INDEX) = 0
```

```
c--
c--      if negative, set sign bit and take absolute value
c--
        IF (IWORD .LT. 0) KBIT(1) = 1
        IF (IWORD .EQ. -32767) GOTO 330
        IWORD = IABS(IWORD)
c--
c--      derive binary equivalent
c--
        DO 10 INDEX = 16,1,-1
          KBIT(INDEX) = MOD(IWORD,2)
          IWORD = IWORD/2
          IF (IWORD .EQ. 0 .AND. KBIT(1) .EQ. 1) GOTO 15         !negative
  10      IF (IWORD .EQ. 0 .AND. KBIT(1) .EQ. 0) GOTO 330         !positive
c--
c--      take 1's complement if negative
c--
  15    DO 20 INDEX = 16,2,-1
  20      KBIT(INDEX) = -KBIT(INDEX) + 1
c--
c--      now take 2's complement
c--
        KBIT(16) = KBIT(16) + 1
        DO 25 INDEX  = 16,2,-1
          IF (KBIT(INDEX) .LE. 1) GOTO 330
          KBIT(INDEX-1) = KBIT(INDEX-1) + 1
  25      KBIT(INDEX) = 0
c--
c--     output binary equivalent
c--
330     CONTINUE
c--
c--     output as you would view them
c--
D30    TYPE 31, IWORD,KBIT
D31    FORMAT (1X,'VALUE',X,I3,' BINARY:  ',4(X,4I1))
c--
c--     convert to DEC format for DAC
c--
        DO 200,I=1,16
200     IBIT(I)=KBIT(17-I)
D       TYPE 400,(J,IBIT(J),J=16,1,-1)
D400    FORMAT (X,I2,')))',2X,I3)
c--
c--      we're done
c--
        RETURN
        END
```

```
      SUBROUTINE BOUCHN(IJK,MFLAG)
c +------------------------------------------------------------------------+
c | Program :    BOULD.REL                                                  |
c | Module  :    BOUCHN.FOR                                                 |
c | Function:    SENDS THE DESIRED DECIMAL CHANNEL NUMBER TO THE SCANNER    |
c |              AFTER CONVERTING IT TO TWO DISTINCT OCTAL NUMBERS          |
c |                                                                         |
c +------------------------------------------------------------------------+
c--
c-----------------------------------------------------------------------
c--   definition of variables in this subroutine
c--
c--   IJK - decimal channel number
c--   MFLAG - error flag if handshake problem
c--   JBIJK - channel designation set to bg program for diagnostics
c--   LCH - decade designation octal number for scanner
c--   LCD - channel designation octal number for scanner
c--   ILCD - handshake counter for decade designation
c--   ILCH - handshake counter for channel designation
c--
c--   NOTE: Each number is converted form a decimal to two different
c--   octal numbers. The first octal number tells the scanner in which
c-    decade the channel is. The second number is the channel in that
c--   decade. For example, decimal channel 32 would be broken down into
c--   decade 3 and channel 2. This will cause channel 32 to be closed on
c--   the scanner.
c--
c--   sequence:  input(32) -) convert [3]   [2]  -)
c--                    send [3] -) send [2] -) 32 closed
c--
c-----------------------------------------------------------------------
c--
c--   control characters
c--
      COMMON /CONTRL/LA,LB,LC,LR,LDLE,LQU,LLAS,LTACT,LLACT,LTAC,
      *LLAC,LI,LF,LCR,LLF,ITRH1,ITRM1,ITRS1,ITRT1,ITRT2,ITRT3
c--
c--   diagnostic block
c--
      LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
      INTEGER*4 I1AREA,I2AREA
      COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
      *LTERM,ITERM,ATIMES,BTIMES,IADPOK,IADPEK,IPHONE,ILDTR,IHDTR,
      *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
      *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
      *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
      *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
      *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
      *JBOVR,JBWAIT,JBSLEP
c-----------------------------------------------------------------------
c--
      MFLAG=0 !init flag
```

```
c--
c--
c--
         CALL IPOKEB(JBDS2,JBCHN)
c--
c--
c--
         CALL SCOPY (' BOUCHN',LEVEL3)
c--
c--
c--
D        CALL PUTSTR(LTERM,' LOCATION: ','0')
D        CALL PUTSTR(LTERM,LEVEL1,' ')
D        CALL PUTSTR(LTERM,LEVEL2,' ')
D        CALL PUTSTR(LTERM,LEVEL3,' ')
D        WRITE (LTERM,*) 'CHANNEL',IJK
c--
c--      converts integer to two different octal numbers
c--      from 0-7
c--
11       LCH=IJK
         JBIJK=IJK
         CALL IPOKEB (JBDCH1,JBIJK)      !poke number at designated address
         LCD=LCH/10
         LCH=LCH-(LCD*10)
c--
c--      convert from integer*2 decimal to octal value equivalent
c--
c--
c--
         LCD=LCD+48       !convert low order to octal
         LCH=LCH+48       !convert high order to octal
         CALL IPOKE (IADPOK,LCD) !poke number at designated address
         CALL ISLEEP (ITRH1,ITRM1,ITRS1,ITRT2)    !wait for results
         ILCD=ILCD+1
         IF (ILCD.GT.10) GO TO 12         !limit handshaking
         IF ((IPEEK(IADPEK)).NE.LCD) GO TO 11    !handshake lcd
c--
c--
c--
         CALL IPOKE (IADPOK,LCH) !poke number at designated address
         CALL ISLEEP (ITRH1,ITRM1,ITRS1,ITRT2)    !sleep
         ILCD=0  !reinitialize counter for this poke sequence
         ILCH=ILCH+1
         IF (ILCH.GT.10) GO TO 12         !limit handshaking
         IF ((IPEEK(IADPEK)).NE.LCH) GO TO 11    !handshake lch
         CALL IPOKEB (JBDCH1,JBIJK)      !channel for audit trail
c--
         RETURN
c--
c--
c--
```

```
c--
c--     error on handshaking
12      MFLAG=1 !set handshaking error flag
        IPOK=(IPEEKB(JBDERR))+JBEHND
        CALL IPOKEB(JBDERR,IPOK)
        RETURN
        END
```

```
      SUBROUTINE TDATE(KYYR,KYMON,KYDAY,KZYR,KZMON,KZDAY,KXDAY)
c +-----------------------------------------------------------------------+
c | Program :    BOULD.REL                                                 |
c | Module  :    BOUDTC.FOR                                                |
c | Function:    DETERMINE DIFFERENCE BETWEEN TWO DATES                    |
c |                                                                        |
c +-----------------------------------------------------------------------+
c |                                                                        |
c |     KYYR - year first date                                             |
c |     KYMON - month first date                                           |
c |     KYDAY - day first date                                             |
c |     KZYR - year second date                                            |
c |     KZMON - month second date                                          |
c |     KZDAY - day second date                                            |
c |     KXDAY - number of days difference between two values               |
c +-----------------------------------------------------------------------+
c--     dimension number of months and the number of days in that month
c--
        DIMENSION KDAYS(12)
        DATA KDAYS/31,28,31,30,31,30,31,31,30,31,30,31/
c--
c--     initialze day differences
c--
        KXDAY=0.
c--
c--     make integer assignments for the months for the different dates
c--     math done in intergers to save storage space
c--
        N1=KYMON
        N2=KZMON
c--
c--     are years the same?
c--     if not branch and calculate difference
c--     if the same continue
c--
        IF (KZYR.NE.KYYR) GOTO 150
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c--
c--     same year
c--
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c--
c--     index is used in computed goto to tell where to go after year is
c--     checked to see if it was a leap year
c--     A is a temporary variable used in leap year calculations
c--
        INDEX=1
        A=KYYR
c--
c--     check for leap year if so use 29 days for FEB.
c--
        GOTO 180
```

```
c--
c--     determine how many months difference ther is
c--     if (1  branch and do calculation
c--     if ) 1 sum months up
c--
200     IF ((KZMON-KYMON).LE.1) GOTO 90 !(= 1MONTH DIFF
c--
c--     month calculations
c--     do calculations from first month +1
c--     to second month -1
c--     for example: if months were May and December you would want to
c--     do the month calculations for the period of June 1 to November 30
c--
        DO 100,L=N1+1,N2-1       !)1 MONTH TOTAL KDAYS
100     KXDAY=KXDAY+KDAYS(L)
c--
c--     calculations for days between months or days in same month
c--
90      IF ((KYMON-KZMON).NE.0) KXDAY=KXDAY+(KDAYS(N1)-KYDAY)   !1 MONTH
        IF ((KYMON-KZMON).EQ.0) KXDAY=KXDAY+(KZDAY-KYDAY)       !SAME MONH
        IF ((KYMON-KZMON).EQ.0) GOTO 300
c--
c--     add this number of days to difference
c--
        KXDAY=KZDAY+KXDAY
c--
c--     return
c--
        GOTO 300
c--
c--     years different  so more calculations need done
c--
c--     make year assignments to integer variables
c--     set index for computed "GOTO" which checks for leap year
c--
150     N1Y=KYYR
        N2Y=KZYR
        INDEX=2
        A=KYYR
c--
c--     go check for leap year
c--
        GOTO 180
c--
c--     determine days until end of month
c--
110     KXDAY=(KDAYS(KYMON)-KYDAY)+KXDAY
c--
c--     if month is December you don't need to sum to end of year
c--
        IF (KYMON.EQ.12.) GOTO 103
c--
```

```
c--     determine days until end of year
c--
        DO 101,L=N1+1,12
        M=L
101     KXDAY=KXDAY+KDAYS(L)
c--
c--     set conditions for leap year check
c--
103     INDEX=3
        A=KZYR
c--
c--     check for leap year
c--
        GOTO 180
c--
c--     is second month January if so you don't need to summ form beginnig
c--     of year
c--
220     IF (KZMON.EQ.1.) GOTO 104
c--
c--     sum from begining of year
c--
        DO 102,L=1,(N2-1)
        M=L
102     KXDAY=KXDAY+KDAYS(L)
c--
c--     add days form beginning of month to diff total
c--
104     KXDAY=KXDAY+KZDAY
c--
c--     if years gt 1 go summ days in years
c--     leap years are calculated also
c--
        IF ((KZYR-KYYR).GT.1) GOTO 175
c--
c--     return
c--
        GOTO 300
c--
c--     determine if leap year if so use 366 and not 365 days for total
c--     days in the year
c--
175     DO 201,K=N1Y+1,N2Y-1
        YR=K
        JYR=YR
        YR=YR/4.
        JYR=JYR/4
        IF (JYR.EQ.YR) KXDAY=KXDAY+366.
        IF (JYR.NE.YR) KXDAY=KXDAY+365.
201     CONTINUE
c--
c--     zzzzz
```

```
c--
300     RETURN
c--
c--     leap year correction for months
c--     determine if years are exacty divisible by 4 with no remainder
c--     if so this is a leap year and February has 29 days
c--
180     YR=A
        JYR=YR
        YR=YR/4.
        JYR=JYR/4
        KDAYS(2)=28
        IF (JYR.EQ.YR) KDAYS(2)=29
c--
c--     zzzzz
c--
        GOTO (200,110,220) INDEX
        END
```

```
      SUBROUTINE DATEI (Y,HYR,HMON,HDAY)
c +---------------------------------------------------------------------+
c | Program :    BOULD.REL                                              |
c | Module  :    BOUDTI.FOR                                             |
c | Function:    CONVERTS REAL*4 DATE INTO 3 REAL*4 VARIABLE            |
c |              ONE FOR YEAR, MONTH, AND DAY                           |
c |                                                                     |
c +---------------------------------------------------------------------+
      HYR=IFIX(Y/10000)
      HMON=IFIX((Y-(HYR*10000))/100)
      HDAY=IFIX(Y-((HYR*10000)+(HMON*100)))
      RETURN
      END
      SUBROUTINE BOUDYR(DOLD,DNEW)
c +----------------------------------------------------------------------+
c | Program :    BOULD.REL                                               |
c | Module  :    BOUDYR.FOR                                              |
c | Function:    RETURNS NEXT DAY OF THE YEAR                            |
c |                                                                      |
c +----------------------------------------------------------------------+
c |  NOTE:                                                               |
c |     returns the next day of the year                                 |
c |       ex: 830223 --> 830224                                          |
c |           830331 --> 830401                                          |
c |                                                                      |
c | DOLD old date                                                        |
c | DNEW next date                                                       |
c |                                                                      |
c |     Dates are stored as real*4 values in the data files              |
c |      so all dates heave been converted to real*4 .                   |
c |                                                                      |
c |      Format: yr;month;day                                            |
c |         830223. is  02/23/83 in Julian form                          |
c |                                                                      |
c |           83 = yr  02 = month   23 = day                             |
c |                                                                      |
c +----------------------------------------------------------------------+
c--
      DIMENSION DAYS(12)
      DATA DAYS/31.,28.,31.,30.,31.,30.,31.,31.,30.,
     *31.,30.,31./
c--
      DNEW=0.
      DNEW=DOLD+1.
c--
c--   check to see if valid date
c--
      XYR=(DNEW/10000.)
      LM=XYR
      XYR=LM
      XMON=((DNEW-(XYR*10000.))/100.)
      LM=XMON
      XMON=LM
      XDAY=DNEW-((XYR*10000.)+(XMON*100.))
      LM=XDAY
      XDAY=LM
c--
c--   check to see leap year
c--
      DAYS(2)=28.      !28 days in february normally
      IF (XMON.NE.2.) GOTO 52 !if not february ignore
      JYR=XYR
      JYR=JYR/4
      AYR=XYR/4.
      IF (JYR.EQ.AYR) DAYS(2)=29.      !leap year 29 days
```

```
c--
c--     ) than days in month correct days and month and year
c--
52      IF (XDAY.GT.DAYS(XMON)) GOTO 10
c--
c--     construct new date
c--
5       DNEW=(XYR*10000.)+(XMON*100.)+(XDAY)
        RETURN
c--
c--     corrections for days,month and year
c--
10      XDAY=1.
        XMON=XMON+1.
        IF (XMON.GT.12.) GOTO 11
        GOTO 5
c--
c--     year correction
c--
11      XYR=XYR+1.
        XMON=1.
        GOTO 5
        END
```

```
c--
190     ICHANT=IGETC()
        IF (ICHANT.LT.0) GOTO 10
c--
c--     find file specified if not there go to error section
c--
        LLOOK=LOOKUP(ICHANT,LFILE)
        IF (LLOOK.LT.0) GOTO 11
c--
c--     is this a read or write????
c--
150     IF (LSTAT.EQ.'WR') GOTO 180
c--
c--     read section  branch if error
c--
        LCOND= IREADW(LWORDS,ARRAY,LBLK,ICHANT)
        IF(LCOND.LT.0) GOTO 12
        GO TO 200
c--
c--     write section branch if error
c--
180     LCOND= IWRITW(LWORDS,ARRAY,LBLK,ICHANT)
        IF(LCOND.LT.0) GOTO 13
c--
c--     if to leave open and no errors return to program
c--
c--     if errors close file anyway
200     IF (LOP.EQ.'OP'.AND.LERR.EQ..FALSE.) RETURN
c--
c--     close section
c--
        CALL CLOSEC(ICHANT)
        CALL IFREEC(ICHANT)
        RETURN
c--
c--     no channels available
c--
10      LERR=-1 !CHANNEL ERROR
        WRITE (LTERM,*)  'RED-C '
        GOTO 200
c--
c--     error on lookup
c--
11      WRITE (LTERM,*) 'RED-L'
        LERR=-2 !LOOKUP ERROR
        IF (LLOOK.EQ.-1) WRITE (LTERM,*) 'L1'
        IF (LLOOK.EQ.-2) WRITE (LTERM,*) 'L2'
        IF (LLOOK.EQ.-3) WRITE (LTERM,*) 'L3'
        IF (LLOOK.EQ.-4) WRITE (LTERM,*) 'L4'
        GO TO 200
c--
c--     error on read
```

```
      SUBROUTINE BOURED(LFILE,ARRAY,LAR,LWORDS,LBLK,LSTAT,LOP,LERR)
c +---------------------------------------------------------------------+
c | Program :     BOULD.REL                                             |
c | Module  :     BOURED.FOR                                            |
c | Function:     READS OR WRITES DATA TO THE SPECIFIED DISK FILE       |
c +---------------------------------------------------------------------+
c |                                                                     |
c |     LFILE   file name       (i*2 array)                             |
c |     LWORDS  length of array in words        (i*2)                   |
c |     ARRAY   array to store or write information     (r*4 array)     |
c |     LAR     array length  (2*lwords) because real*4 array           |
c |     LBLK    block to be read or written to (i*2)                    |
c |     LSTAT   'RE','WR' read or write (i*2)                           |
c |     LOP     'OP','CL','BO' leave open,close only,open/close (i*2)   |
c |     LERR    error flag      (i*2)                                   |
c |                                                                     |
c +---------------------------------------------------------------------+
c--
c--     diagnostic block
c--
      LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
      INTEGER*4 I1AREA,I2AREA
      COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
      *LTERM,ITERM,ATIMES,BTIMES,IADPOK,IADPEK,IPHONE,ILDTR,IHDTR,
      *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
      *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
      *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
      *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
      *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
      *JBOVR,JBWAIT,JBSLEP
c--
c--     dimensions
c--
      DIMENSION LFILE(4),ARRAY(LAR)
c--
c--     if block to be read < 0 return --> invalid record number
c--
      IF (LBLK.LT.0) RETURN
c--
c--     set error flag to .false.
c--
      LERR=.FALSE.    !INITIALIZE ERROR FLAG
c--
c--     bg program diagnostic trace
c--
      CALL IPOKEB(JBDS2,JBRED)
c--
c--     if "close" go to close section
c--
      IF (LOP.EQ.'CL') GOTO 150
c--
c--     get channel and check for errors
```

```
c--
12      WRITE (LTERM,*) 'RED-R'
        LERR=-3 !READ ERROR
        GOTO 15
c--
c--     error on write
c--
13      WRITE (LTERM,*) 'RED-W'
        LERR=-4
15      IF (LCOND.EQ.-1) WRITE (LTERM,*) 'RW1'
        IF (LCOND.EQ.-2) WRITE (LTERM,*) 'RW2'
        IF (LCOND.EQ.-3) WRITE (LTERM,*) 'RW3'
        GOTO 200
        END
```

```
      SUBROUTINE RET(IFLAG)
c +-------------------------------------------------------------------+
c | Program :    BOULD.REL                                            |
c | Module  :    BOURET.FOR                                           |
c | Function:    POKES A CARRIAGE RETURN ("015) TO CLOSE CHANNELS     |
c |                                                                   |
c +-------------------------------------------------------------------+
c--
c--   common block for control characters
c--
      COMMON /CONTRL/LA,LB,LC,LR,LDLE,LQU,LLAS,LTACT,LLACT,LTAC,
      *LLAC,LI,LF,LCR,LLF,ITRH1,ITRM1,ITRS1,ITRT1,ITRT2,ITRT3
c--
c--
c--   diagnostic block
c--
      LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
      INTEGER*4 I1AREA,I2AREA
      COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
      *LTERM,ITERM,ATIMES,BTIMES,IADPOK,IADPEK,IPHONE,ILDTR,IHDTR,
      *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
      *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
      *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
      *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
      *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
      *JBOVR,JBWAIT,JBSLEP
c--
c--   diagnostics to tell bg progarm where you are
c--
      CALL IPOKEB (JBDS2,JBRET)
c--
c--   diagnostic traceback for timing failure
c--
      CALL SCOPY (' RET    ',LEVEL3)
c--
c--   tempory diagnostics for debugging
c--
D     CALL PUTSTR(LTERM,' LOCATION: ','0')
D     CALL PUTSTR(LTERM,LEVEL1,' ')
D     CALL PUTSTR(LTERM,LEVEL2,' ')
D     CALL PUTSTR(LTERM,LEVEL3,' ')
c--
c--   set error flag to 0
c--
      IFLAG = 0
      IRETC=0 !counter for number of times thru
c--
c--   start control sequence and control diagnostic declarations
c--
200   CALL IPOKEB (JBDCTL,30)          !CARRIAGE RETURN
      CALL IPOKE (IADPOK,LCR)          !CARRIAGE RETURN
      CALL ISLEEP (ITRH1,ITRM1,ITRS1,ITRT2)
      IRETC=IRETC+1
      IF (IRETC.GT. 10) GOTO 202
      IF ((IPEEK (IADPEK)).NE.LCR) GO TO 200
C--
      RETURN
202   IFLAG=1
      RETURN
      END
```

```
        SUBROUTINE RLLTI(IFLAG)
c +-----------------------------------------------------------------------+
c | Program :    BOULD.REL                                                |
c | Module  :    BOURLL.FOR                                               |
c | Function:    SETS CONDITIONS SO SCANNER CHANNELS CAN BE CLOSED        |
c |                                                                       |
c +-----------------------------------------------------------------------+
c |                                                                       |
c |     definitions used in this subroutine                               |
c |                                                                       |
c |     iflag= error flag for handshaking problem                         |
c |                                                                       |
c |                                                                       |
c |     ILFAIL= times to go thru subroutine before reporting problem      |
c |     ILFC= counter for interface clear (LC echo LF)                    |
c |     ILFA= counter for attention set (LA echo LF)                      |
c |     ILQU= counter for unlisten (LQU)                                  |
c |     ILTACT= counter for talk addresses controller (LTACT)             |
c |     ILLAS= counter for listen address scanner  (LLAS)                 |
c |     ILC= counter for channel closing  (LC)                            |
c |                                                                       |
c |     NTIMES= times to allow handshaking problems in subsections        |
c |                                                                       |
c +-----------------------------------------------------------------------+
c--
c--     control character common block
c--
        COMMON /CONTRL/LA,LB,LC,LR,LDLE,LQU,LLAS,LTACT,LLACT,LTAC,
        *LLAC,LI,LF,LCR,LLF,ITRH1,ITRM1,ITRS1,ITRT1,ITRT2,ITRT3
c--
c--     diagnostic block
c--
        LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
        INTEGER*4 I1AREA,I2AREA
        COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
        *LTERM,ITERM,ATIMES,BTIMES,IADPOK,IADPEK,IPHONE,ILDTR,IHDTR,
        *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
        *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
        *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
        *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
        *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
        *JBOVR,JBWAIT,JBSLEP
c--
c--     tell bg program where you are
c--
        CALL IPOKEB (JBDS2,JBRLL)
c--
c--     set handshaking limiter
c--

        NTIMES=10       !LIMIT HANDSHAKING ERRORS
c--
```

```
c--     tell internal dignostics where you are in case of failure
c--
        CALL SCOPY (' RLLTI',LEVEL3)
c--
c--     temporary diagnostics for debugging
c--
D       CALL PUTSTR(LTERM,' LOCATION: ','0')
D       CALL PUTSTR(LTERM,LEVEL1,' ')
D       CALL PUTSTR(LTERM,LEVEL2,' ')
D       CALL PUTSTR(LTERM,LEVEL3,' ')
c--
c--     set error flag to 0 and initialze flag counter
c--
        IFLAG=0
        ILFAIL=0
c--
c--     keep track of subroutine errors
c--
1       ILFAIL=ILFAIL+1
        IF (ILFAIL.GT.20) GOTO 200
c--
c--     initial handshake control values
c--
        ILFC=0
        ILFA=0
        ILQU=0
        ILTACT=0
        ILLAS=0
        ILC=0
c--
c--     interface clear
c--
101     CALL IPOKEB (JBDCTL,18)         !DLE
        CALL IPOKE (IADPOK,LDLE)                !DLE
        CALL ISLEEP (ITRH1,ITRM1,ITRS1,ITRT2)
        CALL IPOKEB (JBDCTL,19)         !C
        CALL IPOKE (IADPOK,LC)          !C
        CALL ISLEEP (ITRH1,ITRM1,ITRS1,ITRT3)
        ILFC=ILFC+1
        IF(ILFC.GT.NTIMES) GO TO 1
        IF (IPEEK (IADPEK).NE.LF) GO TO 101     !TEST HANDSHAKE FOR F
c--
c--     remote set
c--
        CALL IPOKEB (JBDCTL,20)         !DLE
        CALL IPOKE (IADPOK,LDLE)                !DLE
        CALL ISLEEP (ITRH1,ITRM1,ITRS1,ITRT1)
        CALL IPOKEB (JBDCTL,21)         !R
        CALL IPOKE (IADPOK,LR)          !R
        CALL ISLEEP (ITRH1,ITRM1,ITRS1,ITRT2)
c--
c--     attention set
```

```
c--
102     CALL IPOKEB (JBDCTL,22)         !DLE
        CALL IPOKE (IADPOK,LDLE)                !DLE
        CALL ISLEEP (ITRH1,ITRM1,ITRS1,ITRT1)
        CALL IPOKEB (JBDCTL,23)         !A
        CALL IPOKE (IADPOK,LA)          !A
        CALL ISLEEP (ITRH1,ITRM1,ITRS1,ITRT2)
        ILFA=ILFA+1
        IF(ILFA.GT.NTIMES) GO TO 1
        IF(IPEEK (IADPEK).NE.LF) GO TO 102      !TEST HANDSHAKE FOR F
c--
c--     unlisten
c--
103     CALL IPOKEB (JBDCTL,24) !?
        CALL IPOKE (IADPOK,LQU) !?
        CALL ISLEEP (ITRH1,ITRM1,ITRS1,ITRT2)
        ILQU=ILQU+1
        IF(ILQU.GT.NTIMES) GO TO 1
        IF(IPEEK (IADPEK).NE.LQU) GO TO 103     !TEST HANDSHAKE FOR ?
c--
c--     talk address controller
c--
104     CALL IPOKEB (JBDCTL,25)         !}
        CALL IPOKE (IADPOK,LTACT)               !}
        CALL ISLEEP (ITRH1,ITRM1,ITRS1,ITRT2)
        ILTACT=ILACT+1
        IF(ILTACT.GT.NTIMES) GO TO 1
        IF(IPEEK (IADPEK).NE.LTACT) GO TO 104   !TEST HANDSHAKE FOR }
c--
c--     listen address scanner
c--
105     CALL IPOKEB (JBDCTL,26)         !)
        CALL IPOKE (IADPOK,LLAS)                !)
        CALL ISLEEP (ITRH1,ITRM1,ITRS1,ITRT2)
        ILLAS=ILLAS+1
        IF(ILLAS.GT.NTIMES) GO TO 1
        IF(IPEEK (IADPEK).NE.LLAS) GO TO 105    !TEST HANDSHAKE FOR )
c--
c--     attention clear
c--
        CALL IPOKEB (JBDCTL,27)         !DLE
        CALL IPOKE (IADPOK,LDLE)                !DLE
        CALL ISLEEP (ITRH1,ITRM1,ITRS1,ITRT1)
        CALL IPOKEB (JBDCTL,28)         !B
        CALL IPOKE (IADPOK,LB)          !B
        CALL ISLEEP (ITRH1,ITRM1,ITRS1,ITRT2)
c--
c--     close following channels
c--
106     CALL IPOKEB (JBDCTL,29)         !C
        CALL IPOKE (IADPOK,LC)          !C
        CALL ISLEEP (ITRH1,ITRM1,ITRS1,ITRT2)
```

```
      ILC=ILC+1
      IF(ILC.GT.NTIMES) GO TO 1
      IF(IPEEK (IADPEK).NE.LC) GO TO 106      !TEST HANDSHAKE FOR C
      RETURN
c--
c--   set error flag
c--
200   IFLAG=0
      RETURN
c--
      END
      SUBROUTINE RTLLI(IFLAG)
c +------------------------------------------------------------------------+
c | Program :    BOULD.REL                                                 |
c | Module  :    BOURTL.FOR                                                |
c | Function:    SET CONDITIONS SO MEASUREMENT SYSTEM SEND DATA            |
c |                                                                        |
c +------------------------------------------------------------------------+
c |                                                                        |
c |     definitions for this subroutine                                    |
c |                                                                        |
c |     counters for control characters                                    |
c |     this limits the amount of handshaking                              |
c |                                                                        |
c |     ILAFAL= counter for number of times subroutine executed with prob|
c |     ILI= LI counter                                                    |
c |     ILA1= first LA counter                                             |
c |     ILA2= second La counter                                            |
c |     ILQU1= first LQU counter                                           |
c |     ILQU2= second LQU counter                                          |
c |     ILTACT= LTACT counter                                              |
c |     ILLTAC= LLTAC counter                                              |
c |     ILLACT= LLACT counter                                              |
c |                                                                        |
c |     NTIMES=number of times to let handshaking problem occur before ac|
c |            is taken                                                    |
c +------------------------------------------------------------------------+
c--
c--     control character common block
c--
        COMMON /CONTRL/LA,LB,LC,LR,LDLE,LQU,LLAS,LTACT,LLACT,LTAC,
        *LLAC,LI,LF,LCR,LLF,ITRH1,ITRM1,ITRS1,ITRT1,ITRT2,ITRT3
c--
c--     diagnostic block
c--
        LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
        INTEGER*4 I1AREA,I2AREA
        COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
        *LTERM,ITERM,ATIMES,BTIMES,IADPOK,IADPEK,IPHONE,ILDTR,IHDTR,
        *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
        *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
        *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
        *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
        *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
        *JBOVR,JBWAIT,JBSLEP
c--
c--     initialize error flag
c--
        IFLAG=0
c--
c--     set diagnostics as to where you are
c--
        CALL IPOKEB(JBDS2,JBRTL)
```

```
c--
c--     initialize hanshaking counter for error flag
c--     this is number of times to let problem occur in each subsection
c--     before action is taken
c--
        NTIMES=10        !LIMIT HANDSHKING ERRORS
c--
c--     diagnostics for program failure
c--
        CALL SCOPY (' RTLLI',LEVEL3)
c--
c--     temporary traceback for debugging
c--
D       CALL PUTSTR(LTERM,' LOCATION: ','0')
D       CALL PUTSTR(LTERM,LEVEL1,' ')
D       CALL PUTSTR(LTERM,LEVEL2,' ')
D       CALL PUTSTR(LTERM,LEVEL3,' ')
c--
c--     counters for handshaking problems
c--

        ILAFAL=0
999     ILI=0   !COUNTER FOR HANDSHAKING PROBLEMS
        ILA1=0
        ILQU1=0
        ILQU2=0
        ILTACT=0
        ILLTAC=0
        ILLACT=0
        ILA2=0
c--
c--     if more than 20 times thru set error flag
c--
        ILAFAL=ILAFAL+1
        IF (ILAFAL.GE.20) GOTO 200
c--
c--
c--     remote set
c--
        CALL IPOKEB (JBDCTL,1)           !DLE
        CALL IPOKE (IADPOK,LDLE)                 !DLE
        CALL ISLEEP (ITRH1,ITRM1,ITRS1,ITRT1)
        CALL IPOKEB (JBDCTL,2)           !R
        CALL IPOKE (IADPOK,LR)           !R
        CALL ISLEEP (ITRH1,ITRM1,ITRS1,ITRT2)
c--
c--     attention set
c--
101     CALL IPOKEB (JBDCTL,3)           !DLE
        CALL IPOKE (IADPOK,LDLE)                 !DLE
        CALL ISLEEP (ITRH1,ITRM1,ITRS1,ITRT1)
        CALL IPOKEB (JBDCTL,4)           !A
```

```
      CALL IPOKE (IADPOK,LA)          !A
      CALL ISLEEP (ITRH1,ITRM1,ITRS1,ITRT2)
      ILA1=ILA1+1
      IF (ILA1.GT.NTIMES) GO TO 999   !LIMIT AMOUNT OF HANDSHAKING
      IF (IPEEK (IADPEK).NE.LF) GO TO 101              !TEST HAND SHAKE F
c--
c--   unlisten
c--
102   CALL IPOKEB (JBDCTL,5)          !?
      CALL IPOKE (IADPOK,LQU)         !?
      CALL ISLEEP (ITRH1,ITRM1,ITRS1,ITRT2)
      ILQU1=ILQU1+1
      IF (ILQU1.GT.NTIMES) GO TO 999  !LIMIT AMOUNT OF HANDSHAKING
      IF (IPEEK (IADPEK).NE.LQU) GO TO 102             !TEST HAND SHAKE ?
c--
c--   talk address controller
c--
103   CALL IPOKEB (JBDCTL,6)          !}
      CALL IPOKE (IADPOK,LTACT)               !}
      CALL ISLEEP (ITRH1,ITRM1,ITRS1,ITRT2)
      ILTACT=ILTACT+1
      IF (ILTACT.GT.NTIMES) GO TO 999 !LIMIT AMOUNT OF HANDSHAKING
      IF (IPEEK (IADPEK).NE.LTACT) GO TO 103           !TEST HANDSHAKE F}
c--
c--   listen address counter
c--
104   CALL IPOKEB (JBDCTL,7)          !*
      CALL IPOKE (IADPOK,LLAC)                !*
      CALL ISLEEP (ITRH1,ITRM1,ITRS1,ITRT2)
      ILLAC=ILLAC+1
      IF (ILLAC.GT.NTIMES) GO TO 999  !LIMIT AMOUNT OF HANDSHAKING
      IF (IPEEK (IADPEK).NE.LLAC) GO TO 104            !TEST HANDSHAKE F*
c--
c--   attention clear
c--
      CALL IPOKEB (JBDCTL,8)          !DLE
      CALL IPOKE (IADPOK,LDLE)                !DLE
      CALL ISLEEP (ITRH1,ITRM1,ITRS1,ITRT1)
      CALL IPOKEB (JBDCTL,9)          !B
      CALL IPOKE (IADPOK,LB)          !B
      CALL ISLEEP (ITRH1,ITRM1,ITRS1,ITRT2)
c--
c--   send initiate
c--
105   CALL IPOKEB (JBDCTL,10)         !I
      CALL IPOKE (IADPOK,LI)          !I
      CALL ISLEEP (ITRH1,ITRM1,ITRS1,ITRT2)
      ILI=ILI+1
      IF (ILI.GT.NTIMES) GO TO 999    !LIMIT AMOUNT OF HANDSHAKING
      IF (IPEEK (IADPEK).NE.LI) GO TO 105              !TEST HANDSHAKE FI
c--
c--   attention set
```

```
c--
107     CALL IPOKEB (JBDCTL,11)          !DLE
        CALL IPOKE (IADPOK,LDLE)                  !DLE
        CALL ISLEEP (ITRH1,ITRM1,ITRS1,ITRT1)
        CALL IPOKEB (JBDCTL,12)          !A
        CALL IPOKE (IADPOK,LA)           !A
        CALL ISLEEP (ITRH1,ITRM1,ITRS1,ITRT2)
        ILA2=ILA2+1
        IF (ILA2.GT.NTIMES) GO TO 999   !LIMIT AMOUNT OF HANDSHAKING
        IF (IPEEK (IADPEK).NE.LF) GO TO 107             !TEST HANDSHAKE FF
c--
c--     unlisten
c--
108     CALL IPOKEB (JBDCTL,13)          !?
        CALL IPOKE (IADPOK,LQU)          !?
        CALL ISLEEP (ITRH1,ITRM1,ITRS1,ITRT2)
        ILQU2=ILQU2+1
        IF (ILQU2.GT.NTIMES) GO TO 999  !LIMIT AMOUNT OF HANDSHAKING
        IF (IPEEK (IADPEK).NE.LQU) GO TO 108            !TEST HAND SHAKE ?
c--
c--     talk address counter
c--
109     CALL IPOKEB (JBDCTL,14)          !J
        CALL IPOKE (IADPOK,LTAC)                  !J
        CALL ISLEEP (ITRH1,ITRM1,ITRS1,ITRT2)
        ILLTAC=ILLTAC+1
        IF (ILLTAC.GT.NTIMES) GO TO 999 !LIMIT AMOUNT OF HANDSHAKING
        IF (IPEEK (IADPEK).NE.LTAC) GO TO 109           !TEST HANDSHAKE FJ
c--
c--     listen address controller
c--
110     CALL IPOKEB (JBDCTL,15)          !}
        CALL IPOKE (IADPOK,LLACT)                 !}
        CALL ISLEEP (ITRH1,ITRM1,ITRS1,ITRT2)
        ILLACT=ILLACT+1
        IF (ILLACT.GT.NTIMES) GO TO 999 !LIMIT AMOUNT OF HANDSHAKING
        IF (IPEEK (IADPEK).NE.LLACT) GO TO 110          !TEST HANDSHAKE F}
c--
c--     attention clear
c--
        CALL IPOKEB (JBDCTL,16)          !DLE
        CALL IPOKE (IADPOK,LDLE)                  !DLE
        CALL ISLEEP (ITRH1,ITRM1,ITRS1,ITRT1)
        CALL IPOKEB (JBDCTL,17)          !B
        CALL IPOKE (IADPOK,LB)           !B
        CALL ISLEEP (ITRH1,ITRM1,ITRS1,ITRT2)
        RETURN
c--
c--     set error flag
c--
200     IFLAG=1
        RETURN
```

END

```
      SUBROUTINE WRITE(SUM,IFLAG)
c +-----------------------------------------------------------------------+
c | Program :    BOULD.REL                                                 |
c | Module  :    BOUWRT.FOR                                                |
c | Function:    READS DATA FROM MEASUREMENT SYSTEM AND CONVERTS IT TO     |
c |              A REAL*4 NUMBER                                           |
c +-----------------------------------------------------------------------+
c |     definitions of values used in this section                        |
c |                                                                       |
c |     IVALUE  - array to store peek values in                           |
c |     VALUE   - array to store converted data values in                 |
c |     NQ      - number of times to peek and poke                        |
c |     SUM     - resultant data value after all conversions              |
c |     EXP     - location of period in string reurned with HP 5300 syste|
c |                                                                       |
c +-----------------------------------------------------------------------+
c--
c--     diagnostic block
c--
        LOGICAL*1 LEVEL1,LEVEL2,LEVEL3,LEVEL4
        INTEGER*4 I1AREA,I2AREA
        COMMON /DIAG/LEVEL1(10),LEVEL2(10),LEVEL3(10),LEVEL4(10),
        *LTERM,ITERM,ATIMES,BTIMES,IADPOK,IADPEK,IPHONE,ILDTR,IHDTR,
        *I1AREA(4),I2AREA(4),SYSTM1,SYSDAT,JBDIO,JBDPGM,JBDS1,JBDS2,
        *JBDCH1,JBDCH2,JBDCTL,JBDERR,JBDAL,JBTMP,JBWND,JBCON,JBPRS,JBFFT,
        *JBERR,JBMAH,JBPMP,JBSTA,JBTIM,JBDK1,JBSLP,JBPKC,JBINT,
        *JBRTL,JBCHN,JBRLL,JBWRT,JBRET,JBRED,JBBIT,JBFAL,JBDISK,JBTT,
        *JBECON,JBEINT,JBETMP,JBEFFT,JBEDAL,JBEHND,JBEADD,JBESHT,JBRUN,
        *JBOVR,JBWAIT,JBSLEP
c--
c--     dimensioning and data declarations for subroutine
c--
        DIMENSION IVALUE(17),VALUE(17)
        DATA NQ/17/EXP/7./
        CALL ISLEEP (0,0,0,15)
        J=0.    !counter for conversion to real
        SUM=0.0 !initalize sum
        IFLAG=0 !INITALIZE ERROR FLAG
c--
c--     diagnostics to tell bg program where you are
c--
        CALL IPOKEB (JBDS2,JBWRT)
c--
c--     diagnostic traceback for timing failure
c--
        CALL SCOPY (' WRITE',LEVEL3)
c--
c--     temporary diagnostics
c--
D       CALL PUTSTR(LTERM,' LOCATION: ','0')
D       CALL PUTSTR(LTERM,LEVEL1,' ')
D       CALL PUTSTR(LTERM,LEVEL2,' ')
```

```
D       CALL PUTSTR(LTERM,LEVEL3,' ')
c--
c--     get the data
c--
        DO 200 I=1,NQ
        CALL ISLEEP (0,0,0,7)
        IVALUE(I)=IPEEK(IADPEK)
200     CALL IPOKE (IADPOK,IVALUE(I))
c--
c----------------------------------------------------------------------------
c--     following calculations dependent upon set up of measuring system
c--     at remote site. in this case we are returning data in Khz
c----------------------------------------------------------------------------
c--
c--     check for validity
c--     if not valid return error flag
c--
c--     location 2  =  F
c--     location 7  =  .
c--     location 13 = E
c--     location 16 = carriage return
c--     location 17 = linefeed
c--
c--     this is defined in measuring system manual for frequency operation
c--     if these values are not in there appropriate places an communicats
c--     error has occurred and the value needs to be resent
c--
D       WRITE (LTERM,39) (IVALUE(I),I=1,17)
        IF (IVALUE(2).NE.70 .AND. IVALUE(7) .NE. 46
        * .AND. IVALUE(13) .NE. 69 .AND. IVALUE(16) .NE. 13
        * .AND. IVALUE(17) .NE. 10) GOTO 201
c--
c--
c--     is it less than 0 or gt 9
c--     decimal 48 = 0
c--     decimal 57 = 9
c--
        DO 101 IZ=2,12
        IF (IVALUE(IZ) .LT. 48 .XOR. IVALUE(IZ) .GT. 57 ) GOTO 101
        J=J+1
        VALUE(J)=IVALUE(IZ)-48
        SUM=SUM+(VALUE(J)*(10.**(EXP-J)))
101     CONTINUE
c--
c--     convert to fregquency
c--
        SUM=SUM/100.
D       WRITE(LTERM,*) 'SUM OUT NORMAL',SUM,IFLAG
        RETURN
c--
c--     handshake error
c--
```

```
201     IFLAG=1
D       WRITE(LTERM,*) 'SUM OUT ERROR',SUM,IFLAG
D       WRITE (LTERM,39) (IVALUE(I),I=1,17)
        RETURN
39      FORMAT (X,'STRING ',17A1)
        END
```

APPENDIX II

MONITORING PROGRAM SOURCE CODE

```
      PROGRAM BOURST
c+---------------------------------------------------------------------+
c| Program :  BOURST.FOR                                                |
c| Module  :  BOURST.FOR                                                |
c| Function:  MAIN DRIVER FOR RESTART PROGRAM                           |
c|            BYTE LOCATIONS ARE CHECKED IN THIS ROUTINE                |
c+---------------------------------------------------------------------+
c+---------------------------------------------------------------------+
c|                                                                      |
c|     Boulder restrt main program for bg testing of foreground status  |
c|                                                                      |
c|      This program checks  on the staus of the foreground program     |
c|        by using byte locations "170 - "177.                          |
c|        Assignments are explained in BOUDAT.FOR (block data routine)  |
c|                                                                      |
c|       See BOURSL.COM for subroutines needed for this program         |
c|                                                                      |
c|                                                                      |
c+---------------------------------------------------------------------+
c--
c--      common for horizontal and vertical positioning of output
c--
         COMMON /SPACE/ MHH1,MHH2,MHH3,MVH1,MVH2,MVH3,
         *MHD1,MHD2,MHD3,MVD1,MVD2,MVD3,MVH4,MVH5,MHH4,MHH5,MHH6,
         *MHD4,MHD5,MHD6
c--
c--      command input dimensioning
c--
         LOGICAL*1 ICHAR,STRING(39)
         EQUIVALENCE (STRING(1),CMD)
c--
c--      label storage
c--
         REAL*4 MAIN(35),OUTPUT(2),SECOND(8),STAT(5),ZTIME(2)
         DIMENSION LS1(5),LS2(35),LS3(8),KL(8),ZOVER(2),MZBLK(4),
         *MCBLK(4),AREA(8)
c--
c--      data for time file read
c--
         DATA MZBLK/3RBOU,3RBPO,3RLLD,3RDAT/LAR/2/LWORDS/4/
c--
c--      data for staus bytes retention
c--
         DATA MCBLK/3RBOU,3RBAV,3RE24,3RDAT/LCAR/8/
         *LCWORD/16/
c--
c--      data for main subroutine
c--
         DATA LS2/0,10,11,12,13,14,15,20,30,40,41,50,60,61,70,
         *71,80,81,82,90,100,110,120,121,130,140,150,151,250,91,
         *92,93,3*0/
c--
```

```
c--   data for secondary subroutine
c--
      DATA LS3/0,10,20,30,40,50,60,70/
c--
c--   program execution status
c--
      DATA LS1/0,1,10,20,30,/IWAIT/0/IHOLD/0/
c--
c--   data for base for peekeb
c--
      DATA IBASE/"170/
c--
c--   data for main subroutine
c--
      DATA MAIN(1)/'    '/MAIN(2)/'DIAL'/MAIN(3)/'TIME'/
      *MAIN(4)/'NUMB'/MAIN(5)/'LOOK'/MAIN(6)/'TEST'/
      *MAIN(7)/'ATCH'/MAIN(8)/'TEMP'/MAIN(9)/'WIND'/
      *MAIN(10)/'COND'/MAIN(11)/'OSC '/MAIN(12)/'PRS '/
      *MAIN(13)/'FFT '/MAIN(14)/'DAC '/MAIN(15)/'ER1 '/
      *MAIN(16)/'ER2 '/MAIN(17)/'MH1 '/MAIN(18)/'MH2 '/
      *MAIN(19)/'MHT '/MAIN(20)/'PMP '/MAIN(21)/'STA '/
      *MAIN(22)/'TIM '/MAIN(23)/'DK1 '/MAIN(24)/'DK2 '/
      *MAIN(25)/'SLP '/MAIN(26)/'PKC '/MAIN(27)/'IN1 '/
      *MAIN(28)/'IN2 '/MAIN(29)/'FAIL'/MAIN(30)/'PDC '/
      *MAIN(31)/'PFT '/MAIN(32)/'PFD '/MAIN(33)/'    '/
      *MAIN(34)/'    '/MAIN(35)/'    '/
c--
c--   data for secondary and control subroutines
c--
      DATA SECOND(1)/'    '/SECOND(2)/'RTL '/
      *SECOND(3)/'CHN '/SECOND(4)/'RLL '/
      *SECOND(5)/'WRT '/SECOND(6)/'RET '/
      *SECOND(7)/'RED '/SECOND(8)/'BIT '/
c--
c--   data for status of program
c--
      DATA STAT/'    ','DATE','WAIT','RUN ','SLEP'/
c--
c--   data for horizontal and vertical positioning
c--   MH-->horizontal    MV-->vertical
c--   @@H -->header   @@D -->data
c--
      DATA MHH1/1/MHH2/35/MHH3/65/MVH1/9/MVH2/11/MVH3/13/
      *MHD1/11/MHD2/44/MHD3/75/MVD1/9/MVD2/11/MVD3/13/
      *MVH4/5/MVH5/7/MHH4/1/MHH5/35/MHH6/65/
      *MHD4/7/MHD5/45/MHD6/70/
c--
c--   data for override
c--
c--
c--   override byye and ouput data device
c--
```

```
      DATA JBDCH2/"175/JBRUN/1/JBDIO/"170/JBTT/10/JBDISK/0/
      DATA ZOVER/' ON ',' OFF'/
      DATA OUTPUT/'DISK    ','TERMINAL'/
c--
c--   phone information for problems
c--
      DATA IPH0,IPH2/"20146,"110176/
c--
c--   set jsw bits
c--
      CALL SCCA(IABORT)       !INHIBIT CONTROL C
      TYPE 1,27        !make sure in ANSI mode
      CALL IPOKE ("044,"10100)
      CALL ISLEEP (0,0,1,0)
c--
c--   init wait -- screen wait for sleep mode
c--    and put location labels on screen
c--
      CALL BOURED(MCBLK,AREA,LCAR,LCWORD,0,'RE','BO',IERR)
      DO 999,L=1,8
      IL=L-1
      ILOOK=IBASE+IL
999   CALL IPOKEB(ILOOK,IFIX(AREA(L)))
4     CALL BOUHDR
      IWAIT=0
c--
c--   time of next run
c--
      CALL BOURED(MZBLK,ZTIME,LAR,LWORDS,0,'RE','BO',IERR)
      CALL VTHVP(MVH4,MHD5)
      TYPE 400,ZTIME(1)       !time of execution
      CALL VTHVP(MVH5,MHD5)
      TYPE 401,ZTIME(2)       !date for execution
c--
c--   look at bytes
c--
5     DO 7,L=1,8
      IL=L-1
      ILOOK=IBASE+IL
7     KL(L)=IPEEKB(ILOOK)
c--
c--   ckeck to see if fg in sleep mode
c--    or if failure occurred
c--
c--   250 timing failure
c--   128 equipment failure
c--
      IF (KL(3).EQ.250.OR.
      *   KL(8).GE.128) GOTO 441
      IF (KL(2) .EQ.LS1(5).AND.IWAIT.EQ.1) GOTO 310
      IF (KL(2).NE.LS1(5).AND. IWAIT.EQ.1) GOTO 4
c--
```

```
c--     get time and date  and put on screen
c--
        TIME=SYSTIM(DATE)
        CALL VTHVP(MVH4,MHD4)
        TYPE 400,TIME
        CALL VTHVP(MVH5,MHD4)
        TYPE 401,DATE
c--
c--     output device
c--
        CALL VTHVP(MVD1,MHD1)
        IF (KL(1).EQ.0) TYPE 30,OUTPUT(1)
        IF (KL(1).EQ.10) TYPE 30,OUTPUT(2)
c--
c--     status byte (date,run wait,sleep)
c--
        CALL VTHVP(MVD1,MHD2)
        DO 35,I=1,5
35      IF (KL(2).EQ.LS1(I)) TYPE 30,STAT(I)
c--
c--     main sub
c--
        CALL VTHVP(MVD2,MHD1)
        DO 31,LK=1,30
31      IF (KL(3).EQ.LS2(LK)) TYPE 30,MAIN(LK)
        IF (KL(3).EQ.250) CALL BOUFLL(1)
c--
c--     secondary sub
c--
        CALL VTHVP(MVD2,MHD2)
        DO 32,LK=1,8
32      IF (KL(4).EQ.LS3(LK)) TYPE 30,SECOND(LK)
c--
c--     channel number
c--
        CALL VTHVP(MVD3,MHD1)
        TYPE 6,KL(5)
c--
c--     override byte
c--
        CALL VTHVP(MVD3,MHD2)
        IF (KL(6).EQ.1) TYPE 44,ZOVER(1)
        IF (KL(6).NE.1) TYPE 44,ZOVER(2)
c--
c--     secondary subroutine location status
c--
        CALL VTHVP(MVD2,MHD3)
        TYPE 6,KL(7)
c--
c--     error byte
c--
        CALL VTHVP(MVD3,MHD3)
        TYPE 6,KL(8)
c--
c--     check to seee if sleep mode
c--
        IF (KL(2).EQ.LS1(5).AND. IWAIT.EQ.0) GOTO 300
c--
c--     has a character been typed ?
c--
        CALL VTHVP(24,IQ+1)
310     ICHAR=ITTINR()
        IF (ICHAR.LT.0) GOTO 5
c--
c--     check for ctrl u
c--
        IF (ICHAR.EQ.21.OR. ICHAR .EQ.127) GOTO 200
c--
c--     check for contrl c      abort --) inhibit
c--
        IF (ICHAR.EQ.3) GOTO 700
c--
c--     location of character in string
c--
102     IQ=IQ+1
        STRING(IQ)=ICHAR
c--
c--     output character
c--
        LK=ITTOUR(ICHAR)
c--
c--     check for 'return' or max 38 characters
c--
        IF (ICHAR.EQ.10.OR. IQ.EQ.38) GOTO 101  !BRANCH ON LF
        GO TO 5
c--
c--     put null byte at end
c--
101     STRING(IQ-1)="000        !PUT NULL BYTE AT END (ASCIZ CODE)
        IQ=0
c--
c--     check to see what was typed
c--
        IF (STRING(1).EQ."000) GOTO 39
        IF (SCOMP('EXIT',STRING).EQ.0) GOTO 40
        IF (SCOMP('QUIT',STRING).EQ.0) GOTO 40
        IF (SCOMP('OVERRIDE',STRING).EQ.0) GOTO 41
        IF (SCOMP('TERMINAL',STRING).EQ.0) GOTO 42
        IF (SCOMP('DISK',STRING).EQ.0) GOTO 43
        IF (SCOMP('STATUS',STRING).EQ.0) GOTO 49
        IF (SCOMP('REFRESH',STRING).EQ.0) GOTO 4
        IF (SCOMP('HELP',STRING).EQ.0) GOTO 160
        IF (SCOMP('HELP/WAIT',STRING).EQ.0) GOTO 161
        IF (SCOMP('START',STRING).EQ.0) CALL BOUFLL(5)
```

```
c--
c--     no command  tell them so
c--
        CALL VTHVP(24,1)
        TYPE 301,"007   !RING BELL IF ERROR
        GO TO 5
c--
c--     set override byte on if off  or  if on set off
c--
41      IF (KL(6).EQ.1) CALL IPOKEB(JBDCH2,0)
        IF (KL(6).NE.1) CALL IPOKEB(JBDCH2,JBRUN)

        GO TO 200
c--
c--     help file
c--
161     IHOLD=1
160     CALL BOUHLP(IHOLD)
        GOTO 4
c--
c--     reinit string to nulls
c--
200     DO 201,I=1,39
201     STRING(I)="000
        IQ=0
        IF (ICHAR.EQ.21.OR.ICHAR.EQ.127) GOTO 39      !clear screen on ctu
        GOTO 5
c--
c--     output to disk
c--
43      CALL IPOKEB (JBDIO,JBDISK)
        GOTO 200
c--
c--     output to terminal
c--
42      CALL IPOKEB(JBDIO,JBTT)
        GOTO 200
c--
c--     if nothing typed erase line and leave screen clear
c--
39      CALL VTHVP(22,1)
        CALL VTED ('E')
        GO TO 5
c--
c--     read disk for information
c--
c--     sleep mode clear screen
c--
300     CALL ISLEEP (0,0,10,0)
        CALL VTHVP(23,1)
        CALL VTED('B')
        CALL ISLEEP(0,0,0,5)
        IWAIT=1
        GOTO 5
c--
c--     reset wait flag for status
c--
49      DO 150,I=1,39
150     STRING(I)="000
        IF (IWAIT.EQ.0) GOTO 5
        GOTO 4
c--
c--     tell them program can not be aborted
c--
700     CALL VTERSE
        TYPE *,'This program cannot be aborted by ctrl C.'
        TYPE *,'If you need help type "HELP".'
        CALL ISLEEP(0,0,2,0)
c--
c--     purge buffer so there are not multiple prints
c--
702     ICHAR=ITTINR()
        IF (ICHAR.GT.0) GOTO 702
        GOTO 4
c--
c--     exit , set margins, and erase screen
c--
40      CALL VTERSE
        IF (SCOMP('EXIT',STRING).EQ.0) GOTO 441
        TYPE *,'ENTER QUIT CODE'
        CALL GETSTR(5,STRING,10,IERR)
        IF (SCOMP('WZ32',STRING).NE.0) GOTO 4
441     CALL VTERSE
        DO 1000,I=1,8
1000    AREA(I)=KL(I)
        IF (KL(3).EQ.250) AREA(3)=-250.
        IF (KL(8).GE.128) AREA(8)=AREA(8)*.001  !CONVERT LT 0
        CALL BOURED(MCBLK,AREA,LCAR,LCWORD,0,'WR','BO',IERR)
        CALL VTTBM(1,24)
        IF (KL(3).EQ.250) CALL BOUFLL(1)
        IF (KL(8).GE.128) CALL BOUFLL(2)
        IF (CMD.EQ.'EXIT') CALL SETCMD('@BFILE')
        CALL EXIT
c-----------------------------------------------------------------------
c--     format statements
c-----------------------------------------------------------------------
1       FORMAT ('$',1A1,'(')
6       FORMAT (I5)
30      FORMAT (X,A4)
44      FORMAT (X,A4)
151     FORMAT (X,'NEXT EXECUTION',10X,'TIME: ',F5.0,5X,'DATE: ',F7.0)
301     FORMAT (X,A1,'TRY AGAIN')
400     FORMAT (X,F5.0)
401     FORMAT (X,F7.0)
```

END

```
      SUBROUTINE BOUHDR
c+--------------------------------------------------------------------+
c| Program :  BOURST.FOR                                               |
c| Module  :  BOUHDR.FOR                                               |
c| Function:  PRINTS HEADER AND LABELS FOR SCREEN                      |
c+--------------------------------------------------------------------+
c--
c--     common for horizontal and vertical positioning of output
c--
      COMMON /SPACE/ MHH1,MHH2,MHH3,MVH1,MVH2,MVH3,
     *MHD1,MHD2,MHD3,MVD1,MVD2,MVD3,MVH4,MVH5,MHH4,MHH5,MHH6,
     *MHD4,MHD5,MHD6
      CALL VTERSE
c--
c--     SET MARGINS
c--
      CALL VTTBM(22,24)
c--
c--     highlite function code
c--
      CALL VTSGR ('U',1,1,1)
c--
c--     boulder status label  double high
c--
      CALL VTHVP(1,8)
      CALL VTDHLT
      TYPE 45
      CALL VTHVP(2,8)
      CALL VTDHLB
      TYPE 45
c--
c--     turn all screen attributes off
c--
      CALL VTSGR('O',1,1,1)
c--
c--     reverse image for labels
c--
      CALL VTSGR ('R',1,1,1)
c--
c--     put on labels
c--
      CALL VTHVP(MVH4,MHH4)
      TYPE *,'Time:'
      CALL VTHVP(MVH4,MHH5)
      TYPE *,'Run Time:'
      CALL VTHVP(MVH5,MHH4)
      TYPE *,'Date:'
      CALL VTHVP(MVH5,MHH5)
      TYPE *,'Run Date:'
      CALL VTHVP (MVH1,MHH1)
      TYPE *,'Output:'
```

```
        CALL VTHVP(MVH1,MHH2)
        TYPE *,'Status:'
        CALL VTHVP(MVH2,MHH1)
        TYPE *,'Main Sub:'
        CALL VTHVP(MVH2,MHH2)
        TYPE *,'Sec Sub:'
        CALL VTHVP(MVH2,MHH3)
        TYPE *,'Location:'
        CALL VTHVP(MVH3,MHH1)
        TYPE *,'Channel:'
        CALL VTHVP(MVH3,MHH2)
        TYPE *,'Override:'
        CALL VTHVP(MVH3,MHH3)
        TYPE *,'Error:'
        CALL VTSGR('O',1,1,1)
        RETURN
c-----------------------------------------------------------------------
c--
c--     format staements
c--
c-----------------------------------------------------------------------
45      FORMAT (X,'Boulder Status Report')
        END
        SUBROUTINE BOUFLL(MM)
c+-----------------------------------------------------------------------+
c| Program :  BOURST.FOR                                                 |
c| Module  :  BOUFLL.FOR                                                 |
c| Function:  EXITS BOURST AND RESATRTS THE FOREGROUND PROGRAM           |
c+-----------------------------------------------------------------------+
c|   NOTE:                                                               |
c|           err=1      timing failure                                   |
c|           err=2      equipment failure                                |
c|           err=5      start foreground                                 |
c|                                                                       |
c|                                                                       |
c|                                                                       |
c+-----------------------------------------------------------------------+
c--
c--     common for horizontal and vertical positioning of output
c--
        COMMON /SPACE/ MHH1,MHH2,MHH3,MVH1,MVH2,MVH3,
        *MHD1,MHD2,MHD3,MVD1,MVD2,MVD3,MVH4,MVH5,MHH4,MHH5,MHH6,
        *MHD4,MHD5,MHD6
c--
c--     reset jsw bits
c--
        CALL IPOKE ("044,"000)
c--
c--     restart foreground
c--
                IF (MM.EQ.5) GOTO 400

c--
c--     exit and restart foreground and background
c--
c--
c--     set screen attributes
c--
401     CALL VTSGR ('R','I','B',0)
c--
c--     main program failure
c--
        CALL VTHVP(MVH3+2,MHH1)
        CALL VTDHLT
        TYPE 10
        CALL VTDHLB
        TYPE 10
c--
c--     output specific problem
c--
        CALL VTHVP(MVH3+4,MHH1)
        CALL VTDHLT
        IF (MM.EQ.1) TYPE 11
        IF (MM.EQ.2) TYPE 12
        CALL VTDHLB
        IF (MM.EQ.1) TYPE 11
        IF (MM.EQ.2) TYPE 12
c--
c--     clear screen attributes and sound bell as to exiting
c--
        CALL VTSGR ('O',0,0,0)
200     DO 100,I=1,15
100     CALL ITTOUR("007)
c--
c--     wait 1 minute and make sure foreground exits properly
c--
        CALL ISLEEP(0,1,0,0)
c--
c--
c--     exit and start restrt command file
c--
400     CALL SETCMD('@RESTRT')
        CALL EXIT
        RETURN
c---------------------------------------------------------------------------
c--
c--     format statements
c--
c---------------------------------------------------------------------------
10      FORMAT (X,'MAIN PROGRAM FAILURE')
11      FORMAT (X,'TIMING FAILURE')
12      FORMAT (X,'EQUIPMENT FAILURE')
        END
```

```
      SUBROUTINE BOUHLP(IHOLD)
c+------------------------------------------------------------------------+
c| Program :  BOURST.FOR                                                  |
c| Module  :  BOUHLP.FOR                                                  |
c| Function:  HELP FILE                                                   |
c+------------------------------------------------------------------------+
      LOGICAL*1 LANS
c--
c--   highlite and underline function code
c--
      CALL VTERSE
      CALL VTWID(132)
      CALL VTSGR ('U','I','R',1)
c--
c--   boulder status label  double high
c--
      CALL VTHVP(1,20)
      CALL VTDWL
      TYPE 45
c--
c--   turn all screen attributes off
c--
      CALL VTSGR('O',1,1,1)
c--
c--   label command section
c--
      CALL VTSGR('R','I',1,1)
      CALL VTHVP(3,17)
      TYPE *,'   COMMANDS   '
      CALL VTSGR('O',0,0,0)
c--
c--   reverse image for labels
c--
      CALL VTSGR ('R',1,1,1)
c--
c--   put on labels
c--
      CALL VTHVP(5,1)
      TYPE *,' DISK     '
      CALL VTHVP(6,1)
      TYPE *,' EXIT     '
      CALL VTHVP(7,1)
      TYPE *,' HELP     '
      CALL VTHVP(8,1)
      TYPE *,' HELP/WAIT'
      CALL VTHVP(9,1)
      TYPE *,' OVERRIDE '
      CALL VTHVP(10,1)
      TYPE *,' QUIT     '
      CALL VTHVP(11,1)
      TYPE *,' REFRESH  '
      CALL VTHVP(12,1)
      TYPE *,' STATUS   '
      CALL VTHVP(13,1)
      TYPE *,' TERMINAL '
c--
c--   turn all screen attributes off
c--
      CALL VTSGR('O',1,1,1)
      CALL VTHVP(5,12)
      TYPE *,'Assigns output to disk file BOU:BOULD.DAT'
      CALL VTHVP(6,12)
      TYPE *,'Exits restart program and starts BOUFIL file program'
      CALL VTHVP(7,12)
      TYPE *,'Invokes Help file  (20 second delay)'
      CALL VTHVP(8,12)
      TYPE *,'Invokes Help file  (indefinite delay)'
      CALL VTHVP(9,12)
      TYPE *,'Sets override byte (toggle on/off)'
      CALL VTHVP(10,12)
      TYPE *,'Exits restart program and returns to monitor'
      CALL VTHVP(11,12)
      TYPE *,'Refreshes screen'
      CALL VTHVP(12,12)
      TYPE *,'Gives status of program during sleep mode'
      CALL VTHVP(13,12)
      TYPE *,'Output to terminal'
c--
c--   label command section
c--
      CALL VTSGR('R','I',1,1)
      CALL VTHVP(3,86)
      TYPE *,'   SCREEN   '
      CALL VTSGR('O',0,0,0)
c--
c--   reverse image for labels
c--
      CALL VTSGR ('R',1,1,1)
c--
c--   put on labels
c--
      CALL VTHVP(5,66)
      TYPE *,' CHANNEL: '
      CALL VTHVP(6,66)
      TYPE *,' DATE:    '
      CALL VTHVP(7,66)
      TYPE *,' ERROR:   '
      CALL VTHVP(8,66)
      TYPE *,' LOCATION:'
      CALL VTHVP(9,66)
      TYPE *,' MAIN SUB:'
      CALL VTHVP(10,66)
      TYPE *,' OUTPUT:  '
      CALL VTHVP(11,66)
      TYPE *,' OVERRIDE:'
      CALL VTHVP(12,66)
      TYPE *,' RUN DATE:'
      CALL VTHVP(13,66)
      TYPE *,' RUN TIME:'
      CALL VTHVP(14,66)
      TYPE *,' SEC SUB: '
      CALL VTHVP(15,66)
      TYPE *,' STATUS:  '
      CALL VTHVP(16,66)
      TYPE *,' TIME:    '
c--
c--   put on definitions
c--
      CALL VTSGR('O',0,0,0)
      CALL VTHVP(5,79)
      TYPE *,'Last channel closed'
      CALL VTHVP(6,79)
      TYPE *,'System date (r*4)'
      CALL VTHVP(7,79)
      TYPE *,'Major error code (see BOUDAT.FOR)'
      CALL VTHVP(8,79)
      TYPE *,'Secondary subroutine location'
      CALL VTHVP(9,79)
      TYPE *,'Main subroutine functioning'
      CALL VTHVP(10,79)
      TYPE *,'Output device'
      CALL VTHVP(11,79)
      TYPE *,'Override failure due to equipment problems'
      CALL VTHVP(12,79)
      TYPE *,'Next date program is to run'
      CALL VTHVP(13,79)
      TYPE *,'Next time program is to run'
      CALL VTHVP(14,79)
      TYPE *,'Control subroutine functioning'
      CALL VTHVP(15,79)
      TYPE *,'mode of operation (wait,sleep,run,date failure)'
      CALL VTHVP(16,79)
      TYPE *,'system time (r*4)'
c--
      IF (IHOLD.EQ.1) GOTO 1000
      CALL ISLEEP(0,0,20,0)
      CALL VTWID(80)
      RETURN
1000  IHOLD=0
      CALL VTHVP(23,1)
      TYPE 500
500   FORMAT (X,'To exit hit "RETURN".  No checking is
      * taking place!'$)
      ACCEPT 501,LANS
      CALL VTWID(80)
```

```
        RETURN
c--------------------------------------------------------------------------
c--
c--     format staements
c--
c--------------------------------------------------------------------------
45      FORMAT (X,'   Status Help File   ')
501     FORMAT (1A1)
        END
```

APPENDIX III

FILE MANAGEMENT SOURCE CODES

```
      PROGRAM BOUFIM
C +------------------------------------------------------------------+
C | Program : BOUFIL                                                 |
C | Module  : BOUFIM                                                 |
C | Function: MAIN DRIVER FOR FILE MANIPULATION                      |
C |                                                                  |
C +------------------------------------------------------------------+
c |                                                                  |
c |    nfile is the number of files                                  |
c |    loc is first line output is on                                |
c |    ir is the row number                                          |
c |    ic is the column number                                       |
c +------------------------------------------------------------------+
      LOGICAL*1 FILE(20),STR(4)
      COMMON /FILE/FILE
      LOGICAL *1 ANS,ANS1,OUTP(13),IDAT(9),ITIM(8),LERR,KERR
      INTEGER*2 MABLK(14,4),MABLKL(14),MABLKW(14),LABLK(4),MZBLK(4)
      DIMENSION DATA(60),MLOC(10),AREA(8)
      COMMON /LOCAT/ NFILE,LOC,IR,IC,LIMIT
      COMMON /DATA/ANS,ANS1,OUTP,IDAT,ITIM,LERR,KERR,
      *MABLK,MABLKL,MABLKW,LABLK,DATA,MLOC,IFILE
      EQUIVALENCE (STR(1),ANS)
c--
c--   output file for print
c--
      DATA FILE /'O','U','P',':','D','A','T','A','.','B',
      *'O','U',8*"000/
c--
c--   define file names
c--
      DATA BOUDEV/3RBOU/
      DATA MABLK(1,1)/3RBOU/MABLK(1,2)/3RBRE/
       MABLK(1,3)/3RADD/MABLK(1,4)/3RDAT/
       MABLK(2,1)/3RBOU/MABLK(2,2)/3RBAV/
       MABLK(2,3)/3RE01/MABLK(2,4)/3RDAT/
       MABLK(3,1)/3RBOU/MABLK(3,2)/3RBAV/
       MABLK(3,3)/3RE24/MABLK(3,4)/3RDAT/
       MABLK(4,1)/3RBOU/MABLK(4,2)/3RBAV/
       MABLK(4,3)/3REDL/MABLK(4,4)/3RDAT/
       MABLK(5,1)/3RBOU/MABLK(5,2)/3RBAV/
       MABLK(5,3)/3REDR/MABLK(5,4)/3RDAT/
       MABLK(6,1)/3RBOU/MABLK(6,2)/3RBAV/
       MABLK(6,3)/3REWK/MABLK(6,4)/3RDAT/
       MABLK(7,1)/3RBOU/MABLK(7,2)/3RBAV/
       MABLK(7,3)/3REWR/MABLK(7,4)/3RDAT/
       MABLK(8,1)/3RBOU/MABLK(8,2)/3RBAV/
       MABLK(8,3)/3REMN/MABLK(8,4)/3RDAT/
       MABLK(9,1)/3RBOU/MABLK(9,2)/3RBAV/
       MABLK(9,3)/3REMR/MABLK(9,4)/3RDAT/
       MABLK(10,1)/3RBOU/MABLK(10,2)/3RBPO/
       MABLK(10,3)/3RLLD/MABLK(10,4)/3RDAT/
       MABLK(11,1)/3RBOU/MABLK(11,2)/3RBAC/
       MABLK(11,3)/3RKUP/MABLK(11,4)/3RDAT/
       MABLK(12,1)/3RBOU/MABLK(12,2)/3RBAL/
       MABLK(12,3)/3RLRD/MABLK(12,4)/3RDAT/
       MABLK(13,1)/3RBOU/MABLK(13,2)/3RBAL/
       MABLK(13,3)/3RLRC/MABLK(13,4)/3RDAT/
       MABLK(14,1)/3RBOU/MABLK(14,2)/3RBST/
       MABLK(14,3)/3RATS/MABLK(14,4)/3RDAT/
c--
c--     define file lengths
c--
        DATA MABLKL/1,1,1,90,1,7,1,30,1,1,1,200,1,1/
c--
c--     define word lengths of files
c--
        DATA MABLKW/120,120,16,120,2,120,2,120,2,4,120,120,2,120/
c--
c-----------------------------------------------------------------------
c--
c--     status file name
c--
        DATA MZBLK/3RBOU,3RBAV,3RE24,3RDAT/LCAR/8/LCWORD/16/
        DATA NFILE/14/LOC/4/IJ/40/
        DATA IBASE/"170/PASWRD/'WZ32'/
c--
c--     set no ctrl c
c--
        CALL SCCA(IABORT)
c--
c--     pokeb current staus bytes
c--
        CALL BOURED(MZBLK,AREA,LCAR,LCWORD,0,'RE','BO',IERR)
95      DO 97,L=1,8
        IL=L-1
        ILOOK=IBASE+IL
97      CALL IPOKEB(ILOOK,IFIX(AREA(L)))
c--
c--     ass output device
c--
        CALL ASSIGN (2,FILE)
c--
c--     get date and time
c--
        CALL DATE(IDAT)
        CALL TIME(ITIM)
c--
        IC=IJ+18
c--
        LIMIT=NFILE+LOC          !NO.FILES +LOCATION OF FIRST LINE FOR EXIT
c--
C--     PAINT SCREEN WITH FILE NAMES
c--
31      IFLAG=0 !REINIT FLAG
```

```
        IR=3
        CALL VTERSE
        CALL VTHVP( 1,1)
        CALL VTSGR('R','I',0,0)
        TYPE *,' Type "RETURN" until file found.
        *Then type function code. [ H for help ] '
        CALL VTSGR('O',0,0,0)
        CALL FILELB     !PUT LABELS ON SCREEN
        CALL VTHVP(LOC,1)
c--
c--     convert from radix 50 format
c--
        DO 12, L=1,NFILE
        DO 11,LK=1,4
11      LABLK(LK)=MABLK(L,LK)
c--
        CALL R50ASC(12,LABLK,OUTP)
        CALL VTHVP(L+IR,IJ)
c--
12      TYPE 2,L,(OUTP(I),I=1,12)
2       FORMAT (X,I2,' ',3A1,':',6A1,'.',3A1)
c--
c--     put exit on screen
c--
        CALL VTHVP(L+IR,IJ)
        TYPE 3
c--
c--     figure out which file they want to look at
c--
20      IR=IR+1
        IF (IR.GT.LIMIT) IR =LOC
        CALL VTHVP(IR,IC)
c--
c--     save last location for line erase if necessary
c--
        ITR=IR
c--
c--     get command
c--
        CALL GETSTR(5,STR,4,IERR)
c--
c--     was this command or number
        CALL CONVRT(REAL,IREAL,STR)
        IF (IREAL.EQ..FALSE.) GOTO 99
c--
c--     number out of range correct
c--
        REAL=ABS(REAL)
        IF (REAL.GE.1..AND.REAL.LE.NFILE) GOTO 98
        REAL= 1.
c--
c--     calculate new location  number + place to start -2
```

```
c--     -2 because need to back up one line for last increment at line 20
c--         and one more for next increment at line 20
c--
98      IR=REAL+LOC-2
c--
c--     goto last line and erase to end
c--
        CALL VTHVP(ITR,IC)
        CALL VTEL('E')
        CALL VTHVP(IR,IC)
        GOTO 20
99      IF (ANS .EQ.'H') GOTO 120       !HELP FILE
        IF (ANS .EQ.'P') GOTO 30        !ON P GET FILE
        IF (ANS.EQ.'S') GOTO 30         !ON S READ FILE BUT ONLY WRITE PAT
        IF (ANS.EQ.'I') GOTO 30         !ON W WRITE TO FILE
        IF (ANS.EQ.'C') GOTO 120        !CREATE NEW FILE
        IF (ANS.EQ.'R') GOTO 30         !screen read of one record
        IF (ANS.EQ.'Q') GOTO 1000       !quit
        IF (ANS.EQ.'E') GOTO 1000       !exit and start BOURST
        IF (ANS.EQ.'D') GOTO 30         !DUMP DATA FOR INFORM
        IF ( ANS.EQ.0) GOTO 20  !ON SPACE BRANCH DOWN ONE LINE
        GO TO 31        !SOMETHING ELSE TYPED REPAINT SCREEN
c--
C--     THIS IS THE FILE
c--
30      CALL CLRBOT
        IF (IR.EQ.LIMIT) GOTO 1000
        IFILE=IR-3      !3 FOR IR IS 4 FIRST TIME THROUGH
c--
C--     CONVERT FROM RADIX 50
c--
        DO 111,LK=1,4
111     LABLK(LK)=MABLK(IFILE,LK)
        CALL R50ASC(12,LABLK,OUTP)
        IF (ANS.EQ.'R') GOTO 120        !skip verification can get out ofy
        CALL VTHVP (21,10)
c--
C--     VERIFY FILE NAME
c--
        TYPE 32,(OUTP(I),I=1,12)
32      FORMAT (' IS THIS THE FILE YOU WANT ? ',3A1,':',6A1,'.',
        *3A1,'  [Y/N] ',$)
        ACCEPT 1,ANS1
c--
C--     NOT PROPER FILE CYCLE BACK TO SEARCH
c--
        CALL CLRLIN
        CALL CLRBOT
        IF (ANS1.EQ.'Y') GOTO 120
        GOTO 20
c--
C--     CORRECT FILE READ IT
```

```
c--
120     IF (ANS .EQ. 'P') CALL READF(IFLAG)
        IF (ANS.EQ.'H') CALL BHELP(IFLAG)
        IF (ANS.EQ.'I') CALL WRITE(IFLAG)
        IF (ANS .EQ.'S') CALL PARTL(IFLAG)
        IF (ANS.EQ.'C') CALL NEWDSK(IFLAG)
        IF (ANS.EQ.'D') CALL BOUFID(IFLAG)
        IF (ANS.EQ.'R') CALL SCRNRD(IFLAG)
        IF (IFLAG.EQ.1) GOTO 31
        GOTO 20
1000    CALL VTERSE
        IF (ANS .NE. 'Q' ) GOTO 999
        CALL VTERSE
        TYPE *,'ENTER QUIT CODE'
        CALL IPOKE("044,"10000)
        ACCEPT 101,CODE
101     FORMAT (A4)
        CALL IPOKE("044,"000)
        IF (CODE.NE.PASWRD) GOTO 31
        CALL VTERSE
999     CALL CLOSE(2)
5       DO 7,L=1,8
        IL=L-1
        ILOOK=IBASE+IL
7       AREA(L)=IPEEKB(ILOOK)
        CALL BOURED(MZBLK,AREA,LCAR,LCWORD,0,'WR','BO',IERR)
        IF (ANS.EQ.'E'
        *    .OR.IR.EQ.LIMIT)
        *        CALL SETCMD('@BSTRT')
        CALL EXIT
1       FORMAT (1A1)
3       FORMAT (4X,'EXIT')
        END
```

```
      SUBROUTINE CLRLIN
C-----------------------------------------------------------------------
C-    THIS PROGRAM CLEARS THE END OF THE LINE
C-----------------------------------------------------------------------
      COMMON /LOCAT/ NFILE,LOC,IR,IC,LIMIT
      CALL VTHVP(IR,IC-1)      !BACK UP ON SPACE
      CALL VTEL('E')  !ERASE TO END OF LINE
      RETURN
      END
      SUBROUTINE CLRBOT
c-----------------------------------------------------------------------
c-    clears bottom of screen
c----------------------------------------------------------------------
      CALL VTHVP (20,1)
      CALL VTED ('E') !ERASE TO END OF DISPLAY
      RETURN
      END
```

```
        SUBROUTINE SCRNRD(IFLAG)
C +-------------------------------------------------------------------+
C | Program : BOUFIL                                                  |
C | Module  : BOUFIS                                                  |
C | Function: ROUTINE TO MODIFY DATA IN A DATA FILE                   |
C |           SCREEN DRIVEN ENTRY                                     |
C |                                                                   |
C +-------------------------------------------------------------------+
c--
        LOGICAL *1 ANS,ANS1,OUTP(13),IDAT(9),ITIM(8),LERR,KERR
        INTEGER*2 MABLK(14,4),MABLKL(14),MABLKW(14),LABLK(4)
        DIMENSION DATA(60),MLOC(10)
        COMMON /LOCAT/ NFILE,LOC,IR,IC,LIMIT
        COMMON /DATA/ANS,ANS1,OUTP,IDAT,ITIM,LERR,KERR,
        *MABLK,MABLKL,MABLKW,LABLK,DATA,MLOC,IFILE
c--
        LOGICAL*1 STRING(6)
c--
c--
c--     password for write
c--
        DATA PASWRD/'WZ32'/
c--
c--     offset for screen orgin that is the first vertical line to start n
c--
        DATA IOFST/3/
c--
        RECORD=0        !init record to zero for each file read
        IFLAG=0 !INIT FLAG
        IWC=0   !init write code to off
        IWORDS=MABLKW(IFILE)
        ILEN=MABLKW(IFILE)/2
        IEND=MABLKL(IFILE)
        LSTAT='RE'
        IOP='BO'
100     CALL VTERSE
44      DO 42 I=1,60
42      DATA(I)=0.
        CALL VTHVP(1,1)
        WRITE (7,206) (OUTP(JK),JK=1,12)
c--
c--     if only one record display it
c--
        IF (IEND.NE.1) GOTO 110
        RECORD=IEND
        GOTO 111
C--
110     CALL CLRBOT
        TYPE *,'Record: '
c--
c--     get record number or command
c--
```

```
        CALL VTHVP(20,10)
        CALL GETSTR (5,STRING,5,IERR)
c--
c--     exit on record 0 or 'e'
c--
        IF (STRING(1).EQ.'E'
        * .OR.STRING(1).EQ.'e'
        *  .OR. STRING(1).EQ.'0')
        *    GOTO 1000
c--
c--     increment one record on return
c--
        IF (STRING(1).NE. 0 ) GOTO 300
         RECORD=RECORD+1
         GOTO 10
c--
c--     convert to number
c--
300     CALL CONVRT(RECORD,IREAL,STRING)
        IF (IREAL.EQ..TRUE.) GOTO 10
c--
c--     branch on command
c--
        IF (STRING(1).EQ.'R') ICW=0     !turn off write function
        IF (STRING(1).EQ.'R') GOTO 110  !branch back to question
        IF (STRING(1).NE.'W') GOTO 500
c--
c--     check to see if they know password
c--
        IF (ICW.EQ.0) GOTO 25
c--
c--     tell them that write function is alreay enabled
c--
        CALL VTHVP(23,1)
        CALL VTSGR('I','R',0,0)
        TYPE 507,"007
        CALL VTSGR('O',0,0,0)
        CALL ISLEEP (0,0,2,0)
        GOTO 110
10      IF (RECORD .LT.0 .OR. RECORD.GT.IEND) GOTO 500
111     CALL VTHVP(1,25)
        TYPE 11,RECORD,IEND
        IBLK=RECORD-1
c--
        CALL BOURED(LABLK,DATA,ILEN,IWORDS,IBLK,LSTAT,IOP,LERR)
        IF (LERR.NE.0) GOTO 201
c--
c--
c--
        IHOR=3
        IVERT=1
        DO 210,ML=1,ILEN
```

```
                CALL VTHVP(IHOR,IVERT)
                WRITE (7,205) (ML,DATA(ML))
                IVERT=IVERT+16
                IF (IVERT.LE.70) GOTO 210
                IVERT=1
                IHOR=IHOR+1
210     CONTINUE
        IF (ICW.EQ.0) GOTO 110
c--
c--     section to allow screen writing
c--
310     CALL CLRBOT
        IVERT=1
        TYPE *,'Location: '
        CALL VTHVP(20,12)
        ACCEPT 312,XLOC
312     FORMAT (F10.0)
        IF (XLOC .LT. 0. .OR. XLOC .GT. ILEN) GOTO 310
        IF (XLOC.EQ.0) GOTO 350
        IXLOC=XLOC
c--
c--     calculate screen coordinates
c--
        IHOR=((IXLOC-1)/5)+IOFST
c--
c--     get value
c--
        CALL VTHVP(20,30)
        TYPE *,'Value: '
        CALL VTHVP(20,40)
        ACCEPT 312,DATA(XLOC)
c--
c--     display new value
C--     values 16 and 15 have to do with the spacing in format statement 5
c--     5 is from the number of values on one line of the screen ie. 5
c--
        IVERT=((XLOC-((IHOR-IOFST)*5))*16)-15
        CALL VTHVP(IHOR,IVERT)
        WRITE (7,205) IXLOC,DATA(IXLOC)
        GOTO 310
350     CALL BOURED(LABLK,DATA,ILEN,IWORDS,IBLK,'WR',IOP,LERR)
        IF (LERR.NE.0) GOTO 201
        GOTO 110
1000    IFLAG=1
        RETURN
201     CALL VTERSE
        CALL VTSGR('B','R','I',0)
        TYPE 202,7,(OUTP(KL),KL=1,12)    !RING BELL
        CALL ISLEEP (0,0,5,0)
        CALL VTSGR ('O',0,0,0)
        IFLAG=1 !ERR ON READ
        RETURN
```

```
c--
c--     error on record number
c--
500     CALL VTHVP(23,1)
        CALL VTSGR('I','R',0,0)
        TYPE 502,"007
        CALL VTSGR('O',0,0,0)
        CALL ISLEEP (0,0,2,0)
        GOTO 110
c--
c--     write protection determination
c--
25      CALL VTHVP(21,1)
        TYPE 30
        CALL IPOKE ("44,"10000)
        ACCEPT 101,CODE
        CALL IPOKE ("044,"000)
        IF (CODE.NE.PASWRD) GOTO 505
        ICW=1
        GOTO 110
505     CALL VTHVP(23,1)
        CALL VTSGR('I','R',0,0)
        TYPE 506,"007
        CALL VTSGR('O',0,0,0)
        CALL ISLEEP (0,0,2,0)
        GOTO 110
c--
c--format statements
c--
11      FORMAT (X,'Record:',F4.0,2X,'of',I4,20X,'"E" to exit')
30      FORMAT (//' Enter write code to proceed')
101     FORMAT (A4)
202     FORMAT (X,1A1,'Error on read or write for  ',
        *3A1,':',6A1,'.',3A1)
205     FORMAT ('+',I2,'-) ',F10.2)
206     FORMAT (X,'FILE: ',3A1,':',6A1,'.',3A1)
502     FORMAT (X,1A1,'No such record')
506     FORMAT (X,1A1,'Illegal Password -  write not enabled.')
507     FORMAT (X,1A1,'Write option already enabled.')
        END
```

```
      SUBROUTINE READF(IFLAG)
C +-------------------------------------------------------------------+
C | Program : BOUFIL                                                  |
C | Module  : BOUFIR                                                  |
C | Function: OUTPUTS ALL NONBLANK RECORDS OF A DATA FILE TO OUTPUT   |
C |           DEVICE                                                  |
C +-------------------------------------------------------------------+
c--
c--   sub to read disk
c--
      LOGICAL *1 ANS,ANS1,OUTP(13),IDAT(9),ITIM(8),LERR,KERR
      INTEGER*2 MABLK(14,4),MABLKL(14),MABLKW(14),LABLK(4)
      DIMENSION DATA(60),MLOC(10)
      COMMON /LOCAT/ NFILE,LOC,IR,IC,LIMIT
      COMMON /DATA/ANS,ANS1,OUTP,IDAT,ITIM,LERR,KERR,
      *MABLK,MABLKL,MABLKW,LABLK,DATA,MLOC,IFILE
C--
C--
      IWORDS=MABLKW(IFILE)
      ILEN=MABLKW(IFILE)/2
      IBEGIN=1
      IEND=MABLKL(IFILE)
      LSTAT='RE'
      IOP='BO'
      ITOT=0  !NUMBER OF BLANK RECORDS IN A FILE
      IFLAG=0
79    CALL CLRBOT
      CALL VTHVP(20,1)
      IF (IEND .EQ. 1) GOTO 50
      TYPE 70
70    FORMAT (X,'READ ALL RECORDS ? [Y/N] '$)
      ACCEPT 71,KERR
71    FORMAT (1A1)
      IF (KERR.EQ.'Y') GOTO 50
      CALL CLRBOT
      TYPE 72
72    FORMAT (X,'FIRST RECORD: '$)
      ACCEPT 73,BEGIN
73    FORMAT (F4.0)
      TYPE 74
74    FORMAT (X,'LAST RECORD: '$)
      ACCEPT 73,END
      IF (BEGIN.LE.0) BEGIN=1
      IF (END.GT.IEND) END=IEND
      IF (END.LT.BEGIN ) GOTO 79
      IBEGIN=BEGIN
      IEND=END
50    CALL VTHVP (IR,IC+4)
      CALL VTSGR('R','I',0,0)
      TYPE *,'FILE BEING READ'
c--
C--   NOW READ THE FILE
```

```
C--
C--
        DO 200,IBLOCK=IBEGIN,IEND
        IBLK=IBLOCK-1   !BLOCK 0 FIRST RECORD
        CALL BOURED(LABLK,DATA,ILEN,IWORDS,IBLK,LSTAT,IOP,LERR)
        IF (LERR.NE.0) GOTO 201
        IZERO=0 !TEST FOR BLANK RECORDS
        DO 43,I=1,ILEN
43      IF (DATA(I).EQ.0) IZERO=IZERO+1
        IF (IZERO.EQ.ILEN) ITOT=ITOT+1
        IF (IZERO.EQ.ILEN) GOTO 64
C--
C--
C--
        WRITE (2,206) (OUTP(JK),JK=1,12),IBLK+1,IEND,IDAT,ITIM
206     FORMAT (//,X,'FILE: ',3A1,':',6A1,'.',3A1,4X,'RECORD',I3,
        *' OF',I3,5X,'DATE: ',9A1,4X,'TIME: ',8A1)
        WRITE (2,205) (ML,DATA(ML),ML=1,ILEN)
205     FORMAT (5(X,I3,'--> ',F10.2))
44      DO 42 I=1,60
42      DATA(I)=0.
200     CONTINUE
        GOTO 65
64      CALL VTHVP (22,10)
        TYPE *,' BLANK RECORD READ  NUMBER',IBLOCK
        CALL ISLEEP (0,0,2,0)
65      CALL VTSGR ('O',0,0,0)
        CALL CLRBOT
        CALL CLRLIN
        RETURN
201     CALL VTERSE
        CALL VTSGR('B','R','I',0)
        TYPE 202,7,(OUTP(KL),KL=1,12)    !RING BELL
202     FORMAT (X,1A1,'ERROR ON READ FOR  '3A1,':',6A1,'.',3A1)
        CALL ISLEEP (0,0,5,0)
        CALL VTSGR ('O',0,0,0)
        IFLAG=1 !ERR ON READ
        RETURN
        END
```

```
      SUBROUTINE BOUFID(IFLAG)
C +-----------------------------------------------------------------+
C | Program : BOUFIL                                                |
C | Module  : BOUFID                                                |
C | Function: OUTPUTS ALL NONBLANK RECORDS TO OUTPUT DEVICE         |
C |                                                                 |
C +-----------------------------------------------------------------+
c--
      LOGICAL *1 ANS,ANS1,OUTP(13),IDAT(9),ITIM(8),LERR,KERR
      INTEGER*2 MABLK(14,4),MABLKL(14),MABLKW(14),LABLK(4)
      DIMENSION DATA(60),MLOC(10)
      COMMON /LOCAT/ NFILE,LOC,IR,IC,LIMIT
      COMMON /DATA/ANS,ANS1,OUTP,IDAT,ITIM,LERR,KERR,
     *MABLK,MABLKL,MABLKW,LABLK,DATA,MLOC,IFILE
C--
C--
      IWORDS=MABLKW(IFILE)
      ILEN=MABLKW(IFILE)/2
      IBEGIN=1
      IEND=MABLKL(IFILE)
      LSTAT='RE'
      IOP='BO'
      ITOT=0  !NUMBER OF BLANK RECORDS IN A FILE
      IFLAG=0
79    CALL CLRBOT
50    CALL VTHVP (IR,IC+4)
      CALL VTSGR('R','I',0,0)
      TYPE *,'FILE BEING READ'
c--
C--   NOW READ THE FILE
c--
c--
      WRITE (2,206) (OUTP(JK),JK=1,12),IDAT,ITIM
206   FORMAT (//,X,'FILE: ',3A1,':',6A1,'.',3A1,4X,
     *'DATE: ',9A1,4X,'TIME: ',8A1)
      DO 200, IBLOCK=IBEGIN,IEND
      IBLK=IBLOCK-1   !BLOCK 0 FIRST RECORD
      CALL BOURED(LABLK,DATA,ILEN,IWORDS,IBLK,LSTAT,IOP,LERR)
      IF (LERR.NE.0) GOTO 201
      IZERO=0 !TEST FOR BLANK RECORDS
      DO 43,I=1,ILEN
43    IF (DATA(I).EQ.0) IZERO=IZERO+1
      IF (IZERO.EQ.ILEN) ITOT=ITOT+1
      IF (IZERO.EQ.ILEN) GOTO 64
c--
c--
c--
      WRITE (2,205) (DATA(ML),ML=1,ILEN)
205   FORMAT (X,5F10.2)
44    DO 42 I=1,60
42    DATA(I)=0.
200   CONTINUE
```

```
      GOTO 65
64    CALL VTHVP (22,10)
      TYPE *,' BLANK RECORD READ  NUMBER',IBLOCK
      CALL ISLEEP (0,0,2,0)
65    CALL VTSGR ('O',0,0,0)
      CALL CLRBOT
      CALL CLRLIN
      RETURN
201   CALL VTERSE
      CALL VTSGR('B','R','I',0)
      TYPE 202,7,(OUTP(KL),KL=1,12)    !RING BELL
202   FORMAT (X,1A1,'ERROR ON READ FOR  '3A1,':',6A1,'.',3A1)
      CALL ISLEEP (0,0,5,0)
      CALL VTSGR ('O',0,0,0)
      IFLAG=1 !ERR ON READ
      RETURN
      END
      SUBROUTINE BHELP(IFLAG)
C +---------------------------------------------------------------------+
C | Program : BOUFIL                                                    |
C | Module  : BOUFIH                                                    |
C | Function: HELP FILE FOR BOUFIL FILE MANAGEMENT SYSTEM               |
C |                                                                     |
C +---------------------------------------------------------------------+
C-
      LOGICAL*1 FILE(20)
      COMMON /FILE/FILE
      LOGICAL*1 LANS,IEQU(4)
      EQUIVALENCE (FILE(1),IEQU(1))
      IFLAG=1
      CALL VTERSE
      CALL VTSGR('R','I',0,0)
      CALL VTHVP(1,20)
      TYPE 10
10    FORMAT (X,'  HELP FILE FOR BOUFIL  ')
      CALL VTSGR('O',0,0,0)
      CALL VTHVP(3,50)
      TYPE 90,IEQU
90    FORMAT (X,'Output device - ',4A1)
      CALL VTHVP(6,1)
      TYPE 80
80    FORMAT (X,'C',T10,'Create new data file')
      TYPE 22
22    FORMAT (X,'D',T10,'Dumps data to DATA.BOU for Inform APPEND')
      TYPE 21
21    FORMAT (X,'E',T10,'Exits and starts BOURST monitor program')
      TYPE 20
20    FORMAT (X,'H',T10,'Help file')
      TYPE 70
70    FORMAT (X,'I',T10,'Reinitializes specified file -
      *Entry code required')
      TYPE 50
50    FORMAT (X,'P',T10,'Reads file specified (all data) -
      * Output data to DATA.BOU')
      TYPE 60
60    FORMAT (X,'Q',T10,'Quit and returns to monitor')
      TYPE 30
30    FORMAT (X,'R',T10,'Reads file specified displays results
      *on screen')
      TYPE 40
40    FORMAT (X,'S',T10,'Reads selected locations on file specified -
      * Output to DATA.BOU')
      CALL VTHVP(20,5)
      CALL VTSGR('R',0,0,0)
      TYPE 200
200   FORMAT (' NOTE: A quick way to get to a particular file is
      * to type the file number.')
      CALL VTSGR('O',0,0,0)
      CALL VTSGR('R',I,0,0)
      CALL VTHVP(23,25)
      TYPE 500
500   FORMAT (X,'To exit hit "RETURN".')
      CALL VTSGR('O',0,0,0)
      ACCEPT 501,LANS
501   FORMAT (1A1)
      RETURN
      END
      SUBROUTINE WRITE(IFLAG)
C +------------------------------------------------------------+
C | Program : BOUFIL                                           |
C | Module  : BOUFIW                                           |
C | Function: INITIALIZES ALL FILE RECORDS TO BLANK            |
C |                                                            |
C +------------------------------------------------------------+
c--
      LOGICAL *1 ANS,ANS1,OUTP(13),IDAT(9),ITIM(8),LERR,KERR
      INTEGER*2 MABLK(14,4),MABLKL(14),MABLKW(14),LABLK(4)
      DIMENSION DATA(60),MLOC(10)
      COMMON /LOCAT/ NFILE,LOC,IR,IC,LIMIT
      COMMON /DATA/ANS,ANS1,OUTP,IDAT,ITIM,LERR,KERR,
     *MABLK,MABLKL,MABLKW,LABLK,DATA,MLOC,IFILE
c--
c--
c--
      IFLAG=0 !INIT FLAG
      CALL VTHVP(20,1)
25    TYPE 30
30    FORMAT (//' ENTER COPY CODE TO PROCEED')
      CALL IPOKE ("44,"10000)
      ACCEPT 101,CODE
      CALL IPOKE ("044,"000)
      IF (CODE.NE.'WZ32') GOTO 500
101   FORMAT (A4)
      CALL CLRBOT
50    CALL VTHVP (IR,IC+4)
      CALL VTSGR('R','I',0,0)
      TYPE *,'FILE INITIALIZATION'
c--
C--   NOW READ THE FILE
c--
      IWORDS=MABLKW(IFILE)
      ILEN=MABLKW(IFILE)/2
      IBEGIN=1
      IEND=MABLKL(IFILE)
      LSTAT='WR'
      IOP='BO'
44    DO 42 I=1,60
42    DATA(I)=0.
c--
      DO 200,IBLOCK=IBEGIN,IEND
      IBLK=IBLOCK-1   !BLOCK 0 FIRST RECORD
      CALL BOURED(LABLK,DATA,ILEN,IWORDS,IBLK,LSTAT,IOP,LERR)
      IF (LERR.NE.0) GOTO 201
c--
c--
c--
200   CONTINUE
      CALL VTHVP (22,10)
      TYPE *,' INITIALIZATION COMPLETE'
```

```
        CALL ISLEEP (0,0,2,0)
        CALL VTSGR ('O',0,0,0)
        CALL CLRBOT
        CALL CLRLIN
        RETURN
201     CALL VTERSE
        CALL VTSGR('B','R','I',0)
        TYPE 202,7,(OUTP(KL),KL=1,12)    !RING BELL
202     FORMAT (X,1A1,'ERROR ON WRITE FOR  '3A1,':',6A1,'.',3A1)
        CALL ISLEEP (0,0,5,0)
        CALL VTSGR ('O',0,0,0)
        IFLAG=1 !ERR ON READ
        RETURN
500     IFLAG=1
        CALL CLRBOT
        TYPE 555,"007
555     FORMAT(X,1A1,'INVALID ID')
        CALL ISLEEP (0,0,2,0)
        IFLAG=1
        RETURN
        END
```

```
      SUBROUTINE PARTL (IFLAG)
C +------------------------------------------------------------------+
C | Program : BOUFIL                                                 |
C | Module  : BOUFIP                                                 |
C | Function: READS AND OUTPUTS ONLY SPECIFIC SECTIONS OF EACH RECORD|
C |           OF A FILE                                              |
C |                                                                  |
C +------------------------------------------------------------------+
c--
c--
      LOGICAL *1 ANS,ANS1,OUTP(13),IDAT(9),ITIM(8),LERR,KERR
      INTEGER*2 MABLK(14,4),MABLKL(14),MABLKW(14),LABLK(4)
      DIMENSION DATA(60),MLOC(10)
      COMMON /LOCAT/ NFILE,LOC,IR,IC,LIMIT
      COMMON /DATA/ANS,ANS1,OUTP,IDAT,ITIM,LERR,KERR,
      *MABLK,MABLKL,MABLKW,LABLK,DATA,MLOC,IFILE
c--
c--
c--
      IFLAG=0 !INIT FLAG
c--
c--   file information
c--
      IWORDS=MABLKW(IFILE)
      ILEN=MABLKW(IFILE)/2
      IBEGIN=1
      IEND=MABLKL(IFILE)
      LSTAT='RE'
      IOP='BO'
      CALL CLRBOT
c--
c--   get number of locations
c--
57    TYPE 55
55    FORMAT (' Number of locations: ',$)
      ACCEPT 56,DONE
      IF (DONE.EQ.0) GOTO 66
      IF (DONE.GT.10) GOTO 57
      IDONE=DONE
      CALL VTHVP(20,1)
      CALL VTED('E')
      DO 59,KL=1,IDONE
70    CALL CLRBOT
      TYPE 60,KL,IDONE
      ACCEPT 56,DONE
      IF (DONE.EQ.0.) GOTO 70
      IF (DONE.GT.ILEN) GOTO 70
59    MLOC(KL)=DONE
      CALL CLRBOT
50    CALL VTHVP (IR,IC+4)
      CALL VTSGR('R','I',0,0)
      TYPE *,'File being read'
```

```
c--
c--     output locations read at top
c--
        WRITE (2,72) "014,(MLOC(K),K=1,IDONE)
c--
c--
c--     now read the file
c--
        DO 200,IBLOCK=IBEGIN,IEND
        IBLK=IBLOCK-1   !BLOCK 0 FIRST RECORD
        CALL BOURED(LABLK,DATA,ILEN,IWORDS,IBLK,LSTAT,IOP,LERR)
        IF (LERR.NE.0) GOTO 201
        IZERO=0 !TEST FOR BLANK RECORDS
        DO 43,I=1,ILEN
43      IF (DATA(I).EQ.0) IZERO=IZERO+1
        IF (IZERO.EQ.ILEN) GOTO 65
c--
c--
c--
51      WRITE (2,207) (DATA(MLOC(KL)),KL=1,IDONE)
44      DO 42 I=1,60
42      DATA(I)=0.
200     CONTINUE
        GOTO 66
65      CALL VTHVP (22,10)
        TYPE *,' Blank records read  number: ',IBLOCK
        CALL ISLEEP (0,0,2,0)
66      CALL VTSGR ('O',0,0,0)
        CALL CLRBOT
        CALL CLRLIN
        RETURN
201     CALL VTERSE
        CALL VTSGR('B','R','I',0)
        TYPE 202,7,(OUTP(KL),KL=1,12)   !RING BELL
        CALL ISLEEP (0,0,5,0)
        CALL VTSGR ('O',0,0,0)
        IFLAG=1 !ERR ON READ
        RETURN
c-----------------------------------------------------------------------
c--
c--     format statements
c--
c-----------------------------------------------------------------------
56      FORMAT (F5.0)
60      FORMAT(' Location:',I2,X,'of',X,I2,X,':',X,$)
72      FORMAT (1A1,/,'Locations read ',10I4,//)
207     FORMAT (X,10F10.2)
202     FORMAT (X,1A1,'ERROR ON READ FOR  '3A1,':',6A1,'.',3A1)
        END
```

```
      SUBROUTINE NEWDSK(IFLAG)
C +---------------------------------------------------------------+
C | Program : BOUFIL                                              |
C | Module  : BOUFIN                                              |
C | Function: THIS PROGRAM CREATES A FILE OF SPECIFIED LENGTH     |
C |           AND NUMBER OF WORDS AS SPECIFIED.                   |
C |                                                               |
C +---------------------------------------------------------------+
c--
c--
c--   dimension statements
c--
      DIMENSION DATA(125)
      INTEGER*2 DBLK(4),CBLK(4)
      REAL*4 IDNAM
      LOGICAL*1 FILE(12),DEV(4),ANSW(6),QUEST(4)
c--
c--   data statements
c--
c--   array initialization of values
c--
      DATA DATA/125*0./CBLK/4*0/DEV/4*"000/
c--
c--   set flag so screen repainted
c--
      IFLAG=1
c--
c--   make sure plenty of queue elements
c--
      CALL IQSET(5)    !MAKE SURE THEIR ARE ENOUGH QUEUE ELEMENTS
c--
c--   start program with explaination
c--
      CALL VTERSE
      CALL VTHVP(1,1)
      TYPE 300
400   IBLK=0  !SET INITIAL BLOCK NUMBER TO 0
c--
c--   get device name and convert to radix 50
c--
      CALL VTHVP(5,1)
      CALL VTED('E')
      TYPE 301
      TYPE 302
      CALL GETSTR (7,DEV,4)
      IF (DEV(1).EQ.0) GOTO 899        !HAVE TO GET DEVICE
      CALL IRAD50 (4,DEV,IDNAM)
c--
c--   check to see if loaded
c--
      CALL IDSTAT (IDNAM,CBLK)         !MAKE SURE DEVICE HANDLER IS LOADD
      IF (CBLK(3).EQ.0)GO TO 500
```

```
c--
c--     get file name
c--
315     TYPE 306
        CALL GETSTR (7,ANSW,6)
        IF (ANSW(1).EQ.0) GOTO 899
c--
c--     assign device naem to file name
c--
        DO 305,I=1,3
305     FILE(I)=DEV(I)
c--
c--     get 6 character file name
c--
        DO 307,I=1,6
        J=I+3
307     FILE(J)=ANSW(I)
c--
c--     assign "dat" as file type
c--
        FILE(10)='D'
        FILE(11)='A'
        FILE(12)='T'
c--
c--     convert to radix 50 format
c--
        CALL IRAD50 (12,FILE,DBLK)
c--
c--     allocate channel
c--
        ICHAN=IGETC()    !GET A CHANNEL
        IF (ICHAN.LT. 0) GOTO 901
c--
c--     information needed to define size of file
c--
c--
c--     how mant blocks long is this file?
c--
        TYPE 310
        ACCEPT 311,BLOCK
        IF (BLOCK.EQ.0) GOTO 899
        NBLOCK=BLOCK
c--
c--     get array length (real*4 value)
c--      real *4 files need 2*words because each value is 4 bytes or 2 wos
c--       an integer is one word - 2 bytes
c--
        TYPE 313
        ACCEPT 314,WORDS
        IF (WORDS.EQ.0) GOTO 899
        NWORDS=WORDS*2.
c--
```

```
c--     see if such a file already exists. if so stop. if not proceed
c--
        ITEST=LOOKUP(ICHAN,DBLK)
        IF(ITEST.EQ.NBLOCK)GO TO 505
c--
c--     create a file, associate it with channel ichan, call it the name
c--      refered to in dlbk, and make it nblock long
c--
c--     if a problem (abnormal return) arises stop the program
c--      failure may be due to trying to enter an existing
c--       file
c--
        ITEST=IENTER(ICHAN,DBLK,NBLOCK)
        IF (ITEST.LT.0) GO TO 510
c--
c--     fill the fill with numbers in data array
c--
        DO 20 I=1,NBLOCK
c--
c--     write to disk nwords of data array on block iblk
c--     on the file associated with channel ichan
c--
        ICODE=IWRITW(NWORDS,DATA,IBLK,ICHAN)
        IF (ICODE.LT.0) GOTO 900
c--
c--     increment block number and do it again
c--
        IBLK=IBLK+1
20      CONTINUE
c--
c--     close the channel and make the file permanent
c--
        CALL CLOSEC (ICHAN)
c--
c--     free the channel
c--
        CALL IFREEC (ICHAN)
c--
c--     build more files?
c--
899     TYPE 320
        CALL GETSTR (7,QUEST,3)
        IF(SCOMP('YES',QUEST,IVAL).EQ.0) GO TO 400
        IF(SCOMP('Y',QUEST,IVAL).EQ.0) GO TO 400
c--
c--     done then exit
c--
1000    RETURN
c--
c--     error on write
c--
900     TYPE *,'ERROR ON WRITE'
```

```
        GOTO 899
c--
c--     no free channels
c--
901     TYPE *,'NO CHANNELS AVAILABLE'
        GOTO 899
c--
c--     device handler error
c--
500     ITEST=IFETCH(IDNAM)
        IF(ITEST.EQ.0) GO TO 315
        IF(ITEST.EQ.1) TYPE 501
        IF(ITEST.EQ.2) TYPE 502
        IF (ITEST.EQ.3) TYPE 503
        TYPE *, 'ERROR LOADING DEVICE HANDLER'
        GOTO 899
c--
c--     file already exists with that structure
c--
505     TYPE 506
        GO TO 899
c--
c--     error on file entry file may already exist
c--     but with different structure
c--
510     IF (ITEST.EQ.-1) TYPE 511
        IF (ITEST.EQ.-2) TYPE 512
        IF (ITEST.EQ.-3) TYPE 513
        IF (ITEST.EQ.-4) TYPE 514
c--
c--     tell them to enter new file name
c--
        TYPE 515
        GO TO 899
c--
c--
c--
c-------------------------------------------------------------------------
c--
c--     FORMAT STATEMENTS
c--
c---------------------------------------------------------------------------
300     FORMAT (' THIS PROGRAM INITIALIZES A DATA FILE ON THE SPECIFIED',,
        1' DEVICE ALLOTING IT THE THE ARRAY LENGTH YOU STATE ',/)
301     FORMAT (' ON WHICH DEVICE DO YOU WANT THE FILE TO BE LOCATED ?')
302     FORMAT (' FOR EXAMPLE: RK0,DX0,RK1,RK2,DX1--  '$)
306     FORMAT (/,' WHAT IS THE SIX CHARACTER NAME OF YOUR FILE ? ')
310     FORMAT (/,' HOW MANY BLOCKS LONG DO YOU WANT THE FILE ? '$)
311     FORMAT (F4.0)
313     FORMAT (/,' WHAT IS THE LENGTH OF THE ARRAY ? (1 TO 125) ' $)
314     FORMAT (F4.0)
320     FORMAT (//,' DO YOU WANT TO BUILD ANY MORE FILES ?  '$)
```

```
501     FORMAT(' DEVICE NAME SPECIFIED DOES NOT EXIST')
502     FORMAT (' NOT ENOUGH MEMOREY TO LOAD HANDLER')
503     FORMAT (' NO HANDLER EXISTS ON THE SYSTEM DEVICE')
506     FORMAT (/,' A FILE BY THAT NAME AND SIZE ALREADY EXISTS.',/
        1' ENTER A NEW FILE NAME.')
511     FORMAT (' FILE NAME OR CHANNEL ALREADY IN USE')
512     FORMAT(' NOT ENOUGH FREE SPACE EXISTS')
513     FORMAT (' DEVICE ALREADY IN USE')
514     FORMAT (' FILE SEQUENCE NUMBER ALREADY IN USE')
515     FORMAT (' ENTER A NEW FILE NAME WHEN REQUESTED')
        END
```

```
      SUBROUTINE FILELB
C +--------------------------------------------------------------------+
C | Program : BOUFIL                                                   |
C | Module  : BOUFIF                                                   |
C | Function: LIST FUNCTIONS OF FILES ON SCREEN                        |
C |                                                                    |
C +--------------------------------------------------------------------+
      COMMON /LOCAT/ NFILE,LOC,IR,IC,LIMIT
      CALL VTHVP (LOC,1)
      CALL VTSGR('R','U',0,0)
      TYPE *,' Most current reading                   '
      TYPE *,' Accumulator for hourly average         '
      TYPE *,' Status bytes for BOURST                '
      TYPE *,' Daily average                          '
      TYPE *,' Record number for daily average        '
      TYPE *,' Data from previous 7 days              '
      TYPE *,' Record number for previous 7 days      '
      TYPE *,' Previous 30 days                       '
      TYPE *,' Record number for previous 30 days     '
      TYPE *,' Next time program is to run            '
      TYPE *,' Backup for current reading             '
      TYPE *,' Last 200 readings                      '
      TYPE *,' Record number for last 200 readings    '
      TYPE *,' Setpoints and timers                   '
      CALL VTSGR('O',0,0,0)
      RETURN
      END
```

```
        SUBROUTINE CONVRT(REALNO,IREAL,STRING)
C +-----------------------------------------------------------------+
C | Program : BOUFIL                                                |
C | Module  : CONVRT                                                |
C | Function: ROUTINE TO CONVERT LOGICAL*1 STRING TO REAL*4         |
C |                                                                 |
C +-----------------------------------------------------------------+
C |       realno    is the real number returned (r*4)               |
C |       ireal     flag that this is actually a real number (i*2)  |
C |                  .true.    real number                          |
C |                  .false.    character string                    |
C |       string    is the character string from the program        |
C |       ilen      is the length of the string array               |
c +-----------------------------------------------------------------+
c-
        REAL*8 DREAL,ANUM(30),AML,EXP
        LOGICAL*1 STRING,IM
        DIMENSION IM(13),STRING(1)
        DATA IM/'+','-','.','0','1','2','3','4','5','6','7','8','9'/
        SIGN=1. !+/- SIGN INITIALIZATION
        IPD=0 !LOCATION OF '.' IN STRING
        IPDT=.FALSE.      !'.' DETECTION FLAG TRUE IF '.'
        IREAL=.FALSE.     !REAL NUMBER FLAG
        REALNO=0.         !INITIALZE REAL NUMBER TO 0.
        DREAL=0.
c-
c-      --is string only a null if so return
c-
        IF (STRING(1).EQ.0) RETURN
c-
c-      --Check for numerical characters in string. If all are not numeril
c-      --return and specify that this is a character string.
c-
        I=0
10      I=I+1
        INUM=.FALSE.      !flag for whether number or character
c-
c-      is string only a null if so branch
c-
        IF (STRING(I).EQ."0) GOTO 26
        L=0      !initialize counter
20      L=L+1
        IF (STRING (I).EQ.IM(L)) INUM=.TRUE.
        IF (INUM.EQ..TRUE.) GOTO 15
        IF (L.LE.13) GOTO 20
        RETURN  !return on .false. --) character not number
c-
c-      --make number equal to specified array location
c-      --for example location 5 would yield number 1  "5-4=1"
c-      --the first four locations are for the special characters and "0"
c-
15      ANUM(I)=L-4
```

```
c-
c-      --check to see if '+' is in first array location only
c-
        IF (STRING(1).EQ.IM(1)) SIGN =.1
        IF (I.NE.1.AND.STRING(I).EQ.IM(1)) RETURN
c-
c-      --check for '-',in first array location only
c-      -- sign is made to be -.1 instead of -1. because the minus sign
c-      --always occupies the first array location of the string and this
c-      --compinsate for something other than a number in that location.
c-
        IF (STRING(I).EQ.IM(2)) SIGN=-.1
        IF (I.NE.1.AND.STRING(I).EQ.IM(2)) RETURN
c-
c-      --check for '.' in array. if none branch
c-
        IF (STRING(I).NE.IM(3)) GOTO 25
c-
c-      --has this been the only occurance of a '.' if no -> return to prm
c-
        IF (IPDT.EQ..TRUE.) RETURN
c-
c-      --make pointer equal to location of the '.'
c-
        IPD=I
c-
c-      --set flag to show that a '.' was detected
c-
        IPDT=.TRUE.
25      GOTO 10
c-
c-      --now that the testing is over with lets see what the number reals
c-
26      I=I-1   !take away one for loc null byte
        IREAL=.TRUE.    !this is a real number
c-
c-
c-
        IF (IPD.EQ.0) IPD=I+1   !put '.' at end of number
        EXP=IPD !location of '.' forconversion by power of 10.
        AML=1   !start with location 1 and raise to highest power
        DO 100 M=1,I
c-
c-      this step ignores the first three values in the IM array
c-      that is the following symbols '+ - .'
c-
        IF (ANUM(M).LT.0.) GOTO 100
c-
        AML=AML+1.
c-
        DREAL=DREAL+(ANUM(M)*(10.**(EXP-AML)))
```

```
c-
100     CONTINUE
        DREAL=DREAL*SIGN
        REALNO=DREAL
        RETURN
        END
```

APPENDIX IV

VIDEO CONTROL

```
        SUBROUTINE VTDHLB
c+------------------------------------------------------------------------+
c| Program :  BOURST,BOUFIL                                               |
c| Module  :  VTHDLB.FOR                                                  |
c| Function:  CAUSES BOTTOM HALF OF LINE TO BE DOUBLE HIGH                |
c+------------------------------------------------------------------------+
C-      MUST BE USED WITH VTDHLB TO SET TOP HALF
C-      THE SAME SEQUENCE MUST BE SENT TO BOTH SUBROUTINES
C-      COMMAND SEQUENCE IS:
C-       ESC  4
C-      ESC=ESCAPE      [ OCTAL 33 ("33) OR DECIMAL 27]
C-
10      TYPE 100,"33
100     FORMAT ('$',A1,' 4')
        RETURN
        END
```

```
      SUBROUTINE VTDHLT
c+-----------------------------------------------------------------------+
c| Program :  BOURST,BOUFIL                                              |
c| Module  :  VTDHLT.FOR                                                 |
c| Function:  CAUSES TOP HALF OF LINE TO BE DOUBLE HIGH                  |
c+-----------------------------------------------------------------------+
C-
C-      MUST BE USED WITH VTDHLB TO SET BOTTOM HALF
C-      THE SAME SEQUENCE MUST BE SENT TO BOTH SUBROUTINES
C-      COMMAND SEQUENCE IS:
C-       ESC  3
C-      ESC=ESCAPE      [ OCTAL 33 ("33) OR DECIMAL 27]
C-
10      TYPE 100,"33
100     FORMAT ('$',A1,' 3')
        RETURN
        END
      SUBROUTINE VTED(IVALUE)
c+------------------------------------------------------------------------+
c| Program :  BOURST.FOR                                                   |
c| Module  :  VTED.FOR                                                     |
c| Function:  ROUTINE TO ERASE SECTIONS OF SCREEN DISPLAY                  |
c+------------------------------------------------------------------------+
C-
C-      COMMAND SEQUENCE IS:
C-      ESC [ PS J
C-      WHERE PS IS ONE OF THE FOLLOWING
C-      0 ERASE FROM ACTIVE POSITION TO END OF DISPLAY
C-      1 ERASE FROM BEGINNING OF DISPLAY TO ACTIVE POSITION
C-      2 ERASE ALL OF DISPLAY
C-      THE FOLLOWINF VALUES WILL BE USED TO DESIGNATE PS
C-      'E' (END) FOR 0 DECIMAL 69
C-      'B' (BEGINNING) FOR 1   DECIMAL 66
C-      'A' (ALL) FOR 2         DECIMAL 65
C-      EXAMPLE FORMAT: CALL VTEL ('A')
C-
        IF (IVALUE.EQ.69) TYPE 10,"33,0
        IF (IVALUE.EQ.66) TYPE 10,"33,1
        IF (IVALUE.EQ.65) TYPE 10,"33,2
10      FORMAT ('$',A1,'[',I1,'J')
        RETURN
        END
        SUBROUTINE VTERSE
c+-----------------------------------------------------------------------+
c| Program :  BOURST.FOR                                                 |
c| Module  :  VTERSE.FOR                                                 |
c| Function:  ERASES SCREEN AND RETURNS TO HOME POSITION                 |
c+-----------------------------------------------------------------------+
C-
C-      ERASE SCREEN AND RETURNS TO HOME POSITION
C-      COMMAND SEQUENCE IS:
C-       ESC [;H -RETURN TO HOME POSITION
C-      ESC [2J -ERASE ALL OF SCREEN
C-      ESC=ESCAPE      [ OCTAL 33 ("33) OR DECIMAL 27]
10      TYPE 100,"33,"133,"073,"110,"33,"133,"062,"112
100     FORMAT ('$',8A1)
        RETURN
        END
        SUBROUTINE VTHVP(ILINE,ICOL)
c+------------------------------------------------------------------------+
c| Program :  BOURST.FOR                                                  |
c| Module  :  VTHVP.FOR                                                   |
c| Function:  HORIZONTAL AND VERTICAL POSITIONING                         |
c+------------------------------------------------------------------------+
C-
C-      HORIZONTAL AND VERTICAL POSITION
C-      COMMAND SEQUENCE IS:
C-      ESC [ PL;PC f
C-      WHERE PL = LINE POSITION AND PC = COLUMN POSITION
C-      ILINE = PL       ICOL=PC
C-      ANY INTEGER MAY BE USED FOR ILINE OR ICOL
C-      SCREEN SIZE IS EITHER 24X80 OR 24X132 DEPENDING ON COLUMN MODE SET
C-      EXAMPLE FORMAT: CALL VTHVP(10,35)
C-      THIS WOULD GOTO LINE 10 AND COLUMN 35
C-
        IF (ILINE.LE.0) ILINE=1
        IF (ICOL.LE.0) ICOL=1
        IF (ILINE.GT.24) ILINE=24
        IF (ICOL.GT.132) ICOL=132
        IF (ILINE.LT.10.AND.ICOL.LT.10) TYPE 10,"33,ILINE,ICOL
        IF (ILINE.GE.10.AND.ICOL.LT.10) TYPE 20,"33,ILINE,ICOL
        IF (ILINE.LT.10.AND.(ICOL.GE.10.AND.ICOL.LT.100))
        9 TYPE 30,"33,ILINE,ICOL
        IF (ILINE.GE.10.AND.(ICOL.GE.10.AND.ICOL.LT.100))
        9 TYPE 40,"33,ILINE,ICOL
        IF (ILINE.LT.10.AND.ICOL.GE.100) TYPE 50,"33,ILINE,ICOL
        IF (ILINE.GE.10.AND.ICOL.GE.100) TYPE 60,"33,ILINE,ICOL
10      FORMAT ('$',A1,'[',I1,';',I1,'f')
20      FORMAT ('$',A1,'[',I2,';',I1,'f')
30      FORMAT ('$',A1,'[',I1,';',I2,'f')
40      FORMAT ('$',A1,'[',I2,';',I2,'f')
50      FORMAT ('$',A1,'[',I1,';',I3,'f')
60      FORMAT ('$',A1,'[',I2,';',I3,'f')
        RETURN
        END
        SUBROUTINE VTSGR(IVAL1,IVAL2,IVAL3,IVAL4)
c+-----------------------------------------------------------------------+
c| Program :  BOURST.FOR                                                 |
c| Module  :  VTSGR.FOR                                                  |
c| Function:  SELECTS TYPE OF GRAPHIC RENDITION (REVERSE VIDEO,UNDERLINE)|
c+-----------------------------------------------------------------------+
C-
C-      SELECT GRAPHIC RENDITION
C-      COMMAND SEQUENCE IS:
C-      ESC [ PS m
C-      WHERE PS IS ONE OF THE FOLLOWING
C-      'O' FOR ALL ATRIBUTES OFF - DECIMAL 79
C-      'I' FOR BOLD OR INCREASED INTENSITY - DECIMAL 73
C-      'U' FOR UNDESCORE - DECIMAL 85
C-      'B' FOR BLINK - DECIMAL 66
C-      'R' FOR NEGATIVE OR REVERSE IMAGE - DECIMAL 82
C-      FORMAT IS CALL VTSGR('I','B',,) OR CALL VTSGR('O',,,)
C-      CAN HAVE UP TO 4 ATTRIBUTES AT ONE TIME
C-
        TYPE 10,"33,0
        IF (IVAL1.EQ.79) TYPE 10,"33,0
        IF (IVAL1.EQ.79) GOTO 1000
        IF (IVAL1.EQ.73) TYPE 10,"33,1
        IF (IVAL1.EQ.85) TYPE 10,"33,4
        IF (IVAL1.EQ.66) TYPE 10,"33,5
        IF (IVAL1.EQ.82) TYPE 10,"33,7
        IF (IVAL2.EQ.73) TYPE 10,"33,1
        IF (IVAL2.EQ.85) TYPE 10,"33,4
        IF (IVAL2.EQ.66) TYPE 10,"33,5
        IF (IVAL2.EQ.82) TYPE 10,"33,7
        IF (IVAL3.EQ.73) TYPE 10,"33,1
        IF (IVAL3.EQ.85) TYPE 10,"33,4
        IF (IVAL3.EQ.66) TYPE 10,"33,5
        IF (IVAL3.EQ.82) TYPE 10,"33,7
        IF (IVAL4.EQ.73) TYPE 10,"33,1
        IF (IVAL4.EQ.85) TYPE 10,"33,4
        IF (IVAL4.EQ.66) TYPE 10,"33,5
        IF (IVAL4.EQ.82) TYPE 10,"33,7
10      FORMAT ('$',A1,'[',I1,'m')
1000    RETURN
        END
```

```
      SUBROUTINE VTTBM (ITOP,IBOT)
c+-----------------------------------------------------------------------+
c| Program :  BOURST.FOR                                                 |
c| Module  :  VTTBM.FOR                                                  |
c| Function:  SET TOP AND BOTTOM MARGINS OF SCREEN                       |
c+-----------------------------------------------------------------------+
C-
C-      SET TOP AND BOTTOM MARGINS
C-      COMMAND SEQUENCE IS:
C-      ESC [ PT;PB r
C-      WHERE PT = TOP MARGIN AND PB = BOTTOM MARGIN
C-      WHERE ITOP=PT      IBOT=PB
C-      ANY INTEGER MAY BE USED FOR ITOP OR IBOT
C-      SCREEN SIZE IS 24 LINES
C-      IBOT MUST BE GREATER THAN ITOP
C-      EXAMPLE FORMAT: CALL VTTBM(10,15)
C-      THIS WOULD SET THE MARGINS OR SCROLLING REGION AS LINE 10-LINE 15
C-
        IF (ITOP.GT.24) ITOP=23
        IF (ITOP.LT.1) ITOP=1
        IF (IBOT.GT.24) IBOT=24
        IF (IBOT.LE.1) IBOT=2
        IF (IBOT.LE.ITOP) IBOT=ITOP+1
        IF (ITOP.LT.10.AND.IBOT.LT.10) TYPE 10,"33,ITOP,IBOT
        IF (ITOP.LT.10.AND.IBOT.GE.10) TYPE 50,"33,ITOP,IBOT
        IF (ITOP.GE.10.AND.IBOT.GE.10) TYPE 60,"33,ITOP,IBOT
10      FORMAT ('$',A1,'[',I1,';',I1,'r')
50      FORMAT ('$',A1,'[',I1,';',I2,'r')
60      FORMAT ('$',A1,'[',I2,';',I2,'r')
        RETURN
        END
      SUBROUTINE VTWID(I)
c+-------------------------------------------------------------------------+
c| Program :  BOURST.FOR                                                    |
c| Module  :  VTWID.FOR                                                     |
c| Function:  CHANGES WIDTH OF SCREEN                                       |
c+-------------------------------------------------------------------------+
c--
c--     change from 80-132 columns
c--     format for 80 columns
c--             ESC [?3l          little L
c--     format for 132 columns
c--             ESC [?3h
c--     if no value or wrong value 80 columns default
c--
c--     format:
c--     CALL VTWID(132)           !SCREEN 132 CHARACTERS
c--
        IF (I.EQ.132) TYPE 20,"33
        IF (I.NE.132) TYPE 10,"33
10      FORMAT ('$',A1,'[?3l')
20      FORMAT ('$',A1,'[?3h')
        I=0
        RETURN
        END
      SUBROUTINE VTDWL
c+-------------------------------------------------------------------------+
c| Program :  BOURST.FOR                                                   |
c| Module  :  VTDWL.FOR                                                    |
c| Function:  SET DOUBLE WIDTH LINE                                        |
c+-------------------------------------------------------------------------+
c--
c--     command sequence is:
c--      ESC  6
c--     ESC = escape    [ octal 33 ("33) or decimal 27]
c--
c--     format
c--     CALL VTDWL
c--     TYPE *,'text'
c--
c--     causes "text" to be double width
c--
10      TYPE 100,"33
100     FORMAT ('$',A1,' 6')
        RETURN
        END
      SUBROUTINE VTEL(IVALUE)
C +-------------------------------------------------------------------+
C | Program : BOUFIL                                                  |
C | Module  : VTEL                                                    |
C | Function: ERASES SECTION OF SCREEN LINES                          |
C |                                                                   |
C +-------------------------------------------------------------------+
C-      COMMAND SEQUENCE IS:
C-
C-      ESC [ PS K
C-
C-      WHERE PS IS ONE OF THE FOLLOWING
C-
C-      0 ERASE FROM ACTIVE POSITION TO END OF LINE
C-      1 ERASE FROM BEGINNING OF LINE TO ACTIVE POSITION
C-      2 ERASE ALL OF LINE
C-
C-      THE FOLLOWING VALUES WILL BE USED TO DESIGNATE PS
C-
C-      'E' (END) FOR 0 DECIMAL 69
C-      'B' (BEGINNING) FOR 1   DECIMAL 66
C-      'A' (ALL) FOR 2         DECIMAL 65
C-
C-      EXAMPLE FORMAT: CALL VTEL ('A')
C-
        IF (IVALUE.EQ.69) TYPE 10,"33,0
        IF (IVALUE.EQ.66) TYPE 10,"33,1
        IF (IVALUE.EQ.65) TYPE 10,"33,2
10      FORMAT ('$',A1,'[',I1,'K')
        RETURN
        END
```

APPENDIX V

LINKAGE MAP FILES

```
RT-11 LINK  V06.01D       Load Map            Mon 28-Nov-83 08:43:19
BOULD .REL        Title:  BOULD     Ident:    FORV02
Section    Addr   Size    Global    Value     Global    Value     Global    Value

 . ABS.    000000 001000   (RW,I,GBL,ABS,OVR)
                          $USRSW    000000    $RF2A1    000000    .VIR      000002
                          $NLCHN    000006    $HRDWR    000010    $SYSV$    000012
                          $WASIZ    000152    $LRECL    000210
 $OHAND    001000 000106   (RW,I,GBL,REL,CON)
                          $OVRH     001002    O$READ    001024    O$DONE    001036
                          $ODF1     001102    $ODF2     001104
 $OTABL    001106 000776   (RW,D,GBL,REL,OVR)
 OTS$I     002104 022052   (RW,I,LCL,REL,CON)
                          $$OTSI    002104    $CVTDB    002104    $CVTDI    002104
                          $CVTCB    002120    $CVTCI    002120    CIC$      002132
                          CLC$      002132    CID$      002142    CLD$      002142
                          $DI       002142    $CVTFB    002146    $CVTFI    002146
                          CIF$      002160    CLF$      002160    $RI       002160
                          CIL$      002172    CLI$      002176    $CVTIC    002200
                          $CVTID    002200    CCI$      002212    CDI$      002212
                          $IC       002212    $ID       002212    $CVTIF    002216
                          CFI$      002230    $IR       002230    RCI$      002242
                          GCO$      003256    FCO$      003264    ECO$      003270
                          DCO$      003276    ADF$IS    004220    ADF$PS    004234
                          ADF$MS    004246    $ADDF     004256    ADF$SS    004270
                          $ADR      004270    SUF$IS    004300    SUF$PS    004314
                          SUF$MS    004326    $SUBF     004336    SUF$SS    004350
                          $SBR      004350    DIF$MS    004362    DIF$PS    004366
                          DIF$IS    004374    $DIVF     004402    DIF$SS    004414
                          $DVR      004414    MUF$MS    004426    MUF$PS    004432
                          MUF$IS    004440    $MULF     004446    MUF$SS    004460
                          $MLR      004460    $OTI      004516    $$OTI     004520
                          $SETOP    004742    $$SET     006550    $INITI    007044
                          ASSIGN    007162    NMI$1M    007606    NMI$1I    007620
                          BLE$      007630    BEQ$      007632    BGT$      007640
                          BGE$      007642    BRA$      007644    BNE$      007650
                          BLT$      007652    CAI$      007662    CAL$      007670
                          $CLOSE    007720    CLOSE     010476    CMF$PS    010522
                          CMF$MS    010526    CMF$IS    010542    $CMPF     010550
                          CMF$SS    010562    $CMR      010562    CMF$PI    010574
                          CMF$MI    010600    CMF$II    010610    CMF$SI    010614
                          CMF$PP    010626    CMF$MP    010632    CMF$IP    010642
                          CMF$SP    010646    CMF$PM    010656    CMF$MM    010662
                          CMF$IM    010672    CMF$SM    010676    OCI$      010730
                          ICI$      010736    $ECI      010752    OCO$      011132
                          ICO$      011140    DIM       011336    $DUMPL    011406
                          END$      011534    ERR$      011546    $END      011560
                          $ERR      011576    $OPNER    011620    $CHKER    011656
                          $IOEXI    011702    $EOL      011750    EOL$      011752
                          $ERRTB    012066    $ERRS     012173    EXIT      015734
                          $FCHNL    015740    $FIO      016602    $$FIO     016606
                          $GETFI    017752    IDATE     020010    IFW$      020122
```

```
                        $IFW     020126   $$IFW    020132   IFW$$    020170
                        ILW$     020240   $ILW     020244   TVS$     020364
                        $TVS     020366   MOI$RS   021200   MOL$RS   021200
                        MOI$RM   021204   MOI$RP   021210   MOI$RA   021212
                        MOI$SS   021216   MOL$SS   021216   MOI$SM   021222
                        MOI$SA   021226   MOI$IS   021232   MOL$IS   021232
                        REL$     021232   MOI$IM   021236   MOI$IA   021242
                        MOI$MS   021246   MOI$MM   021252   MOI$MA   021256
                        MOI$0S   021262   MOI$0M   021266   MOI$0A   021272
                        MOI$1S   021276   MOI$1M   021304   MOI$1A   021312
                        ICI$S    021320   ICI$M    021324   ICI$P    021330
                        ICI$A    021332   DCI$S    021336   DCI$M    021342
                        DCI$P    021346   DCI$A    021350   MOL$SM   021354
                        MOL$SA   021360   MOL$MS   021364   MOL$MM   021374
                        MOL$MA   021400   MOL$SP   021404   MOL$PP   021412
                        MOL$MP   021416   MOL$PM   021426   MOL$PS   021434
                        MOL$PA   021440   MOL$IM   021446   MOL$IA   021454
                        MOL$IP   021462   TSL$S    021472   TSL$M    021476
                        TSL$I    021502   TSL$P    021510   $PUTRE   021516
                        RET$L    022024   RET$F    022030   RET$I    022036
                        RET$     022040   $PUTBL   022074   $GETBL   022304
                        $EOFIL   022470   $EOF2    022504   SAVRG$   022524
                        THRD$    022702   $STPS    022704   STP$     022712
                        $STP     022712   FOO$     022716   $EXIT    022736
                        $OTIS    023062   $$OTIS   023064   TAI$     023204
                        $TAI     023204   TAL$     023212   $TAL     023212
                        TAQ$     023220   $TAQ     023220   TAP$     023226
                        $TAP     023226   TAD$     023234   $TAD     023234
                        TAF$     023242   $TAF     023242   TVL$     023464
                        $TVL     023464   TVF$     023472   $TVF     023472
                        TVD$     023500   $TVD     023500   TVQ$     023506
                        $TVQ     023506   TVP$     023514   $TVP     023514
                        TVI$     023522   $TVI     023522   $WAIT    023656
                        $VRINT   023720   SAL$IP   024130   SAL$SP   024132
                        SVL$IP   024136   SVL$SP   024140   SAL$MP   024146
                        SVL$MP   024152
OTS$P    024156 000054   (RW,D,GBL,REL,OVR)
SYS$I    024232 001176   (RW,I,LCL,REL,CON)
                        $CMPLT   024232   CVTTIM   024350   $DIVTK   024412
                        $DIV60   024440   GTIM     024504   ICMKT    024524
                        IPEEK    024556   IPOKE    024566   IPEEKB   024576
                        IPOKEB   024612   IQSET    024622   ITIMER   024720
                        ISCHED   024734   ISLEEP   025006   ISPY     025046
                        $GVAL    025052   LEN      025070   SCOPY    025106
                        $TIME1   025202   $TIME$   025314
USER$I   025430 000000   (RW,I,LCL,REL,CON)
$CODE    025430 002416   (RW,I,LCL,REL,CON)
                        $$OTSC   025430   TIMDIF   026714   SYSTIM   027354
                        PUTSTR   027554
OTS$O    030046 001066   (RW,I,LCL,REL,CON)
                        $$OTSO   030046   $OPEN    030046
SYS$O    031134 000374   (RW,I,LCL,REL,CON)
```

```
 $DATAP   031530 000766   (RW,D,LCL,REL,CON)
 OTS$D    032516 000006   (RW,D,LCL,REL,CON)
                         NHCLN$  032522
 OTS$S    032524 000002   (RW,D,LCL,REL,CON)
                         $AOTS   032524
 SYS$S    032526 000004   (RW,D,LCL,REL,CON)
                         $SYSLB  032526  $LOCK   032530  $CRASH  032531
 $DATA    032532 000106   (RW,D,LCL,REL,CON)
 USER$D   032640 000000   (RW,D,LCL,REL,CON)
 .$$$$.   032640 000000   (RW,D,GBL,REL,OVR)
 BOULD    032640 001764   (RW,D,GBL,REL,OVR)
 CHAN     034624 000140   (RW,D,GBL,REL,OVR)
 CONDCL   034764 000134   (RW,D,GBL,REL,OVR)
 CONTRL   035120 000052   (RW,D,GBL,REL,OVR)
 DIAG     035172 000300   (RW,D,GBL,REL,OVR)
 FILE     035472 000350   (RW,D,GBL,REL,OVR)
 NUMBER   036042 000026   (RW,D,GBL,REL,OVR)
 PROCES   036070 000010   (RW,D,GBL,REL,OVR)
 TIMER    036100 000020   (RW,D,GBL,REL,OVR)
 WAIT     036120 000022   (RW,D,GBL,REL,OVR)
 BITS     036142 000042   (RW,D,GBL,REL,OVR)
 Segment size = 036204 = 7746.  words
Overlay region  000001 Segment 000001
 OTS$I    036206 000000   (RW,I,LCL,REL,CON)
 SYS$I    036206 000000   (RW,I,LCL,REL,CON)
 USER$I   036206 000000   (RW,I,LCL,REL,CON)
 $CODE    036206 001220   (RW,I,LCL,REL,CON)
                         BOUCHK  036206
 OTS$O    037426 000000   (RW,I,LCL,REL,CON)
 SYS$O    037426 000000   (RW,I,LCL,REL,CON)
 $DATAP   037426 000654   (RW,D,LCL,REL,CON)
 OTS$D    040302 000000   (RW,D,LCL,REL,CON)
 OTS$S    040302 000000   (RW,D,LCL,REL,CON)
 SYS$S    040302 000000   (RW,D,LCL,REL,CON)
 $DATA    040302 000032   (RW,D,LCL,REL,CON)
 USER$D   040334 000000   (RW,D,LCL,REL,CON)
 Segment size = 002126 = 555.   words
Overlay region  000001 Segment 000002
 OTS$I    036206 000000   (RW,I,LCL,REL,CON)
 SYS$I    036206 000000   (RW,I,LCL,REL,CON)
 USER$I   036206 000000   (RW,I,LCL,REL,CON)
 $CODE    036206 001212   (RW,I,LCL,REL,CON)
                         BOUCON@ 036206
 OTS$O    037420 000000   (RW,I,LCL,REL,CON)
 SYS$O    037420 000000   (RW,I,LCL,REL,CON)
 $DATAP   037420 000176   (RW,D,LCL,REL,CON)
 OTS$D    037616 000000   (RW,D,LCL,REL,CON)
 OTS$S    037616 000000   (RW,D,LCL,REL,CON)
 SYS$S    037616 000000   (RW,D,LCL,REL,CON)
 $DATA    037616 000026   (RW,D,LCL,REL,CON)
 USER$D   037644 000000   (RW,D,LCL,REL,CON)
 Segment size = 001436 = 399.   words
```

```
Overlay region  000001 Segment 000003
 OTS$I    036206 000000    (RW, I, LCL, REL, CON)
 SYS$I    036206 000000    (RW, I, LCL, REL, CON)
 USER$I   036206 000000    (RW, I, LCL, REL, CON)
 $CODE    036206 001474    (RW, I, LCL, REL, CON)
                          BOUDAC@ 036206
 OTS$O    037702 000000    (RW, I, LCL, REL, CON)
 SYS$O    037702 000000    (RW, I, LCL, REL, CON)
 $DATAP   037702 000476    (RW, D, LCL, REL, CON)
 OTS$D    040400 000000    (RW, D, LCL, REL, CON)
 OTS$S    040400 000000    (RW, D, LCL, REL, CON)
 SYS$S    040400 000000    (RW, D, LCL, REL, CON)
 $DATA    040400 000124    (RW, D, LCL, REL, CON)
 USER$D   040524 000000    (RW, D, LCL, REL, CON)
 Segment size = 002316 = 615.   words
Overlay region  000001 Segment 000004
 OTS$I    036206 000000    (RW, I, LCL, REL, CON)
 SYS$I    036206 000334    (RW, I, LCL, REL, CON)
                          MTATCH  036206  MTGET   036214  MTSTAT  036326
                          MTPRNT  036342  MTDTCH  036350  MTRCTO  036356
                          MTOUT   036364  MTIN    036372  MTSET   036400
 USER$I   036542 000000    (RW, I, LCL, REL, CON)
 $CODE    036542 000624    (RW, I, LCL, REL, CON)
                          DIALT @ 036542
 OTS$O    037366 000000    (RW, I, LCL, REL, CON)
 SYS$O    037366 000000    (RW, I, LCL, REL, CON)
 $DATAP   037366 000420    (RW, D, LCL, REL, CON)
 OTS$D    040006 000000    (RW, D, LCL, REL, CON)
 OTS$S    040006 000000    (RW, D, LCL, REL, CON)
 SYS$S    040006 000000    (RW, D, LCL, REL, CON)
 $DATA    040006 000032    (RW, D, LCL, REL, CON)
 USER$D   040040 000000    (RW, D, LCL, REL, CON)
 Segment size = 001632 = 461.   words
Overlay region  000001 Segment 000005
 OTS$I    036206 000000    (RW, I, LCL, REL, CON)
 SYS$I    036206 000000    (RW, I, LCL, REL, CON)
 USER$I   036206 000000    (RW, I, LCL, REL, CON)
 $CODE    036206 001534    (RW, I, LCL, REL, CON)
                          BOUDK1@ 036206
 OTS$O    037742 000000    (RW, I, LCL, REL, CON)
 SYS$O    037742 000000    (RW, I, LCL, REL, CON)
 $DATAP   037742 000474    (RW, D, LCL, REL, CON)
 OTS$D    040436 000000    (RW, D, LCL, REL, CON)
 OTS$S    040436 000000    (RW, D, LCL, REL, CON)
 SYS$S    040436 000000    (RW, D, LCL, REL, CON)
 $DATA    040436 000064    (RW, D, LCL, REL, CON)
 USER$D   040522 000000    (RW, D, LCL, REL, CON)
 Segment size = 002314 = 614.   words
Overlay region  000001 Segment 000006
 OTS$I    036206 000000    (RW, I, LCL, REL, CON)
 SYS$I    036206 000000    (RW, I, LCL, REL, CON)
 USER$I   036206 000000    (RW, I, LCL, REL, CON)
```

```
  $CODE    036206 002314   (RW,I,LCL,REL,CON)
                             BOUDK2@ 036206
  OTS$O    040522 000000   (RW,I,LCL,REL,CON)
  SYS$O    040522 000000   (RW,I,LCL,REL,CON)
  $DATAP   040522 000660   (RW,D,LCL,REL,CON)
  OTS$D    041402 000000   (RW,D,LCL,REL,CON)
  OTS$S    041402 000000   (RW,D,LCL,REL,CON)
  SYS$S    041402 000000   (RW,D,LCL,REL,CON)
  $DATA    041402 000066   (RW,D,LCL,REL,CON)
  USER$D   041470 000000   (RW,D,LCL,REL,CON)
  Segment size = 003262 = 857.   words
 Overlay region  000001 Segment 000007
  OTS$I    036206 000000   (RW,I,LCL,REL,CON)
  SYS$I    036206 000000   (RW,I,LCL,REL,CON)
  USER$I   036206 000000   (RW,I,LCL,REL,CON)
  $CODE    036206 000504   (RW,I,LCL,REL,CON)
                             BOUER1@ 036206
  OTS$O    036712 000000   (RW,I,LCL,REL,CON)
  SYS$O    036712 000000   (RW,I,LCL,REL,CON)
  $DATAP   036712 000432   (RW,D,LCL,REL,CON)
  OTS$D    037344 000000   (RW,D,LCL,REL,CON)
  OTS$S    037344 000000   (RW,D,LCL,REL,CON)
  SYS$S    037344 000000   (RW,D,LCL,REL,CON)
  $DATA    037344 000012   (RW,D,LCL,REL,CON)
  USER$D   037356 000000   (RW,D,LCL,REL,CON)
  Segment size = 001150 = 308.   words
 Overlay region  000001 Segment 000010
  OTS$I    036206 000000   (RW,I,LCL,REL,CON)
  SYS$I    036206 000000   (RW,I,LCL,REL,CON)
  USER$I   036206 000000   (RW,I,LCL,REL,CON)
  $CODE    036206 001162   (RW,I,LCL,REL,CON)
                             BOUER2@ 036206
  OTS$O    037370 000000   (RW,I,LCL,REL,CON)
  SYS$O    037370 000000   (RW,I,LCL,REL,CON)
  $DATAP   037370 000706   (RW,D,LCL,REL,CON)
  OTS$D    040276 000000   (RW,D,LCL,REL,CON)
  OTS$S    040276 000000   (RW,D,LCL,REL,CON)
  SYS$S    040276 000000   (RW,D,LCL,REL,CON)
  $DATA    040276 000032   (RW,D,LCL,REL,CON)
  USER$D   040330 000000   (RW,D,LCL,REL,CON)
  Segment size = 002122 = 553.   words
 Overlay region  000001 Segment 000011
  OTS$I    036206 000000   (RW,I,LCL,REL,CON)
  SYS$I    036206 000014   (RW,I,LCL,REL,CON)
                             PRINT   036206  RCTRLO  036216
  USER$I   036222 000000   (RW,I,LCL,REL,CON)
  $CODE    036222 000530   (RW,I,LCL,REL,CON)
                             FAIL  @ 036222
  OTS$O    036752 000000   (RW,I,LCL,REL,CON)
  SYS$O    036752 000000   (RW,I,LCL,REL,CON)
  $DATAP   036752 000666   (RW,D,LCL,REL,CON)
  OTS$D    037640 000000   (RW,D,LCL,REL,CON)
  OTS$S   037640 000000   (RW,D,LCL,REL,CON)
  SYS$S   037640 000000   (RW,D,LCL,REL,CON)
  $DATA   037640 000010   (RW,D,LCL,REL,CON)
  USER$D  037650 000000   (RW,D,LCL,REL,CON)
  Segment size = 001442 = 401.   words
 Overlay region  000001 Segment 000012
  OTS$I   036206 000000   (RW,I,LCL,REL,CON)
  SYS$I   036206 000000   (RW,I,LCL,REL,CON)
  USER$I  036206 000000   (RW,I,LCL,REL,CON)
  $CODE   036206 000524   (RW,I,LCL,REL,CON)
                        BOUFFT@ 036206
  OTS$O   036732 000000   (RW,I,LCL,REL,CON)
  SYS$O   036732 000000   (RW,I,LCL,REL,CON)
  $DATAP  036732 000120   (RW,D,LCL,REL,CON)
  OTS$D   037052 000000   (RW,D,LCL,REL,CON)
  OTS$S   037052 000000   (RW,D,LCL,REL,CON)
  SYS$S   037052 000000   (RW,D,LCL,REL,CON)
  $DATA   037052 000016   (RW,D,LCL,REL,CON)
  USER$D  037070 000000   (RW,D,LCL,REL,CON)
  Segment size = 000662 = 217.   words
 Overlay region  000001 Segment 000013
  OTS$I   036206 000000   (RW,I,LCL,REL,CON)
  SYS$I   036206 000120   (RW,I,LCL,REL,CON)
                        IDSTAT  036206  IFETCH  036224
  USER$I  036326 000000   (RW,I,LCL,REL,CON)
  $CODE   036326 000702   (RW,I,LCL,REL,CON)
                        BOUIN1@ 036326
  OTS$O   037230 000000   (RW,I,LCL,REL,CON)
  SYS$O   037230 000000   (RW,I,LCL,REL,CON)
  $DATAP  037230 000772   (RW,D,LCL,REL,CON)
  OTS$D   040222 000000   (RW,D,LCL,REL,CON)
  OTS$S   040222 000000   (RW,D,LCL,REL,CON)
  SYS$S   040222 000000   (RW,D,LCL,REL,CON)
  $DATA   040222 000040   (RW,D,LCL,REL,CON)
  USER$D  040262 000000   (RW,D,LCL,REL,CON)
  Segment size = 002054 = 534.   words
 Overlay region  000001 Segment 000014
  OTS$I   036206 000000   (RW,I,LCL,REL,CON)
  SYS$I   036206 000000   (RW,I,LCL,REL,CON)
  USER$I  036206 000000   (RW,I,LCL,REL,CON)
  $CODE   036206 000730   (RW,I,LCL,REL,CON)
                        BOUIN2@ 036206
  OTS$O   037136 000000   (RW,I,LCL,REL,CON)
  SYS$O   037136 000000   (RW,I,LCL,REL,CON)
  $DATAP  037136 000360   (RW,D,LCL,REL,CON)
  OTS$D   037516 000000   (RW,D,LCL,REL,CON)
  OTS$S   037516 000000   (RW,D,LCL,REL,CON)
  SYS$S   037516 000000   (RW,D,LCL,REL,CON)
  $DATA   037516 000032   (RW,D,LCL,REL,CON)
  USER$D  037550 000000   (RW,D,LCL,REL,CON)
  Segment size = 001342 = 369.   words
 Overlay region  000001 Segment 000015
 OTS$I    036206 000000   (RW,I,LCL,REL,CON)
 SYS$I    036206 000000   (RW,I,LCL,REL,CON)
 USER$I   036206 000000   (RW,I,LCL,REL,CON)
 $CODE    036206 000436   (RW,I,LCL,REL,CON)
                             BOUIN3@ 036206
 OTS$O    036644 000000   (RW,I,LCL,REL,CON)
 SYS$O    036644 000000   (RW,I,LCL,REL,CON)
 $DATAP   036644 000174   (RW,D,LCL,REL,CON)
 OTS$D    037040 000000   (RW,D,LCL,REL,CON)
 OTS$S    037040 000000   (RW,D,LCL,REL,CON)
 SYS$S    037040 000000   (RW,D,LCL,REL,CON)
 $DATA    037040 000366   (RW,D,LCL,REL,CON)
 USER$D   037426 000000   (RW,D,LCL,REL,CON)
 Segment size = 001220 = 328.   words
Overlay region  000001 Segment 000016
 OTS$I    036206 000000   (RW,I,LCL,REL,CON)
 SYS$I    036206 000000   (RW,I,LCL,REL,CON)
 USER$I   036206 000000   (RW,I,LCL,REL,CON)
 $CODE    036206 001374   (RW,I,LCL,REL,CON)
                             BOUMH1@ 036206
 OTS$O    037602 000000   (RW,I,LCL,REL,CON)
 SYS$O    037602 000000   (RW,I,LCL,REL,CON)
 $DATAP   037602 000226   (RW,D,LCL,REL,CON)
 OTS$D    040030 000000   (RW,D,LCL,REL,CON)
 OTS$S    040030 000000   (RW,D,LCL,REL,CON)
 SYS$S    040030 000000   (RW,D,LCL,REL,CON)
 $DATA    040030 000056   (RW,D,LCL,REL,CON)
 USER$D   040106 000000   (RW,D,LCL,REL,CON)
 Segment size = 001700 = 480.   words
Overlay region  000001 Segment 000017
 OTS$I    036206 000000   (RW,I,LCL,REL,CON)
 SYS$I    036206 000000   (RW,I,LCL,REL,CON)
 USER$I   036206 000000   (RW,I,LCL,REL,CON)
 $CODE    036206 001440   (RW,I,LCL,REL,CON)
                             BOUMH2@ 036206
 OTS$O    037646 000000   (RW,I,LCL,REL,CON)
 SYS$O    037646 000000   (RW,I,LCL,REL,CON)
 $DATAP   037646 000314   (RW,D,LCL,REL,CON)
 OTS$D    040162 000000   (RW,D,LCL,REL,CON)
 OTS$S    040162 000000   (RW,D,LCL,REL,CON)
 SYS$S    040162 000000   (RW,D,LCL,REL,CON)
 $DATA    040162 000114   (RW,D,LCL,REL,CON)
 USER$D   040276 000000   (RW,D,LCL,REL,CON)
 Segment size = 002070 = 540.   words
Overlay region  000001 Segment 000020
 OTS$I    036206 000000   (RW,I,LCL,REL,CON)
 SYS$I    036206 000000   (RW,I,LCL,REL,CON)
 USER$I   036206 000000   (RW,I,LCL,REL,CON)
 $CODE    036206 000762   (RW,I,LCL,REL,CON)
                             BOUOSC@ 036206
 OTS$O    037170 000000   (RW,I,LCL,REL,CON)
 SYS$O    037170 000000   (RW,I,LCL,REL,CON)
 $DATAP   037170 000150    (RW,D,LCL,REL,CON)
 OTS$D    037340 000000    (RW,D,LCL,REL,CON)
 OTS$S    037340 000000    (RW,D,LCL,REL,CON)
 SYS$S    037340 000000    (RW,D,LCL,REL,CON)
 $DATA    037340 000022    (RW,D,LCL,REL,CON)
 USER$D   037362 000000    (RW,D,LCL,REL,CON)
 Segment size = 001154 = 310.   words
Overlay region  000001 Segment 000021
 OTS$I    036206 000000    (RW,I,LCL,REL,CON)
 SYS$I    036206 000000    (RW,I,LCL,REL,CON)
 USER$I   036206 000000    (RW,I,LCL,REL,CON)
 $CODE    036206 001556    (RW,I,LCL,REL,CON)
                          BOUMHT@ 036206
 OTS$O    037764 000000    (RW,I,LCL,REL,CON)
 SYS$O    037764 000000    (RW,I,LCL,REL,CON)
 $DATAP   037764 000372    (RW,D,LCL,REL,CON)
 OTS$D    040356 000000    (RW,D,LCL,REL,CON)
 OTS$S    040356 000000    (RW,D,LCL,REL,CON)
 SYS$S    040356 000000    (RW,D,LCL,REL,CON)
 $DATA    040356 000106    (RW,D,LCL,REL,CON)
 USER$D   040464 000000    (RW,D,LCL,REL,CON)
 Segment size = 002256 = 599.   words
Overlay region  000001 Segment 000022
 OTS$I    036206 000000    (RW,I,LCL,REL,CON)
 SYS$I    036206 000000    (RW,I,LCL,REL,CON)
 USER$I   036206 000000    (RW,I,LCL,REL,CON)
 $CODE    036206 001754    (RW,I,LCL,REL,CON)
                          BOUPDC@ 036206
 OTS$O    040162 000000    (RW,I,LCL,REL,CON)
 SYS$O    040162 000000    (RW,I,LCL,REL,CON)
 $DATAP   040162 000576    (RW,D,LCL,REL,CON)
 OTS$D    040760 000000    (RW,D,LCL,REL,CON)
 OTS$S    040760 000000    (RW,D,LCL,REL,CON)
 SYS$S    040760 000000    (RW,D,LCL,REL,CON)
 $DATA    040760 000140    (RW,D,LCL,REL,CON)
 USER$D   041120 000000    (RW,D,LCL,REL,CON)
 Segment size = 002712 = 741.   words
Overlay region  000001 Segment 000023
 OTS$I    036206 000000    (RW,I,LCL,REL,CON)
 SYS$I    036206 000000    (RW,I,LCL,REL,CON)
 USER$I   036206 000000    (RW,I,LCL,REL,CON)
 $CODE    036206 000636    (RW,I,LCL,REL,CON)
                          BOUPFT@ 036206
 OTS$O    037044 000000    (RW,I,LCL,REL,CON)
 SYS$O    037044 000000    (RW,I,LCL,REL,CON)
 $DATAP   037044 000206    (RW,D,LCL,REL,CON)
 OTS$D    037252 000000    (RW,D,LCL,REL,CON)
 OTS$S    037252 000000    (RW,D,LCL,REL,CON)
 SYS$S    037252 000000    (RW,D,LCL,REL,CON)
 $DATA    037252 000032    (RW,D,LCL,REL,CON)
 USER$D   037304 000000    (RW,D,LCL,REL,CON)
 Segment size = 001076 = 287.   words
```

```
Overlay region  000001 Segment 000024
 OTS$I    036206 000000   (RW,I,LCL,REL,CON)
 SYS$I    036206 000000   (RW,I,LCL,REL,CON)
 USER$I   036206 000000   (RW,I,LCL,REL,CON)
 $CODE    036206 000066   (RW,I,LCL,REL,CON)
                           BOUPKC@ 036206
 OTS$O    036274 000000   (RW,I,LCL,REL,CON)
 SYS$O    036274 000000   (RW,I,LCL,REL,CON)
 $DATAP   036274 000056   (RW,D,LCL,REL,CON)
 OTS$D    036352 000000   (RW,D,LCL,REL,CON)
 OTS$S    036352 000000   (RW,D,LCL,REL,CON)
 SYS$S    036352 000000   (RW,D,LCL,REL,CON)
 $DATA    036352 000000   (RW,D,LCL,REL,CON)
 USER$D   036352 000000   (RW,D,LCL,REL,CON)
 Segment size = 000144 = 50.    words
Overlay region  000001 Segment 000025
 OTS$I    036206 000000   (RW,I,LCL,REL,CON)
 SYS$I    036206 000000   (RW,I,LCL,REL,CON)
 USER$I   036206 000000   (RW,I,LCL,REL,CON)
 $CODE    036206 001346   (RW,I,LCL,REL,CON)
                           BOUPMP@ 036206
 OTS$O    037554 000000   (RW,I,LCL,REL,CON)
 SYS$O    037554 000000   (RW,I,LCL,REL,CON)
 $DATAP   037554 000222   (RW,D,LCL,REL,CON)
 OTS$D    037776 000000   (RW,D,LCL,REL,CON)
 OTS$S    037776 000000   (RW,D,LCL,REL,CON)
 SYS$S    037776 000000   (RW,D,LCL,REL,CON)
 $DATA    037776 000034   (RW,D,LCL,REL,CON)
 USER$D   040032 000000   (RW,D,LCL,REL,CON)
 Segment size = 001624 = 458.   words
Overlay region  000001 Segment 000026
 OTS$I    036206 000000   (RW,I,LCL,REL,CON)
 SYS$I    036206 000000   (RW,I,LCL,REL,CON)
 USER$I   036206 000000   (RW,I,LCL,REL,CON)
 $CODE    036206 001456   (RW,I,LCL,REL,CON)
                           BOUPRS@ 036206
 OTS$O    037664 000000   (RW,I,LCL,REL,CON)
 SYS$O    037664 000000   (RW,I,LCL,REL,CON)
 $DATAP   037664 000220   (RW,D,LCL,REL,CON)
 OTS$D    040104 000000   (RW,D,LCL,REL,CON)
 OTS$S    040104 000000   (RW,D,LCL,REL,CON)
 SYS$S    040104 000000   (RW,D,LCL,REL,CON)
 $DATA    040104 000034   (RW,D,LCL,REL,CON)
 USER$D   040140 000000   (RW,D,LCL,REL,CON)
 Segment size = 001732 = 493.   words
Overlay region  000001 Segment 000027
 OTS$I    036206 000000   (RW,I,LCL,REL,CON)
 SYS$I    036206 000000   (RW,I,LCL,REL,CON)
 USER$I   036206 000000   (RW,I,LCL,REL,CON)
 $CODE    036206 000742   (RW,I,LCL,REL,CON)
                           BOUSLP@ 036206   TIMADD   036572
 OTS$O    037150 000000   (RW,I,LCL,REL,CON)
```

```
  SYS$O    037150 000000    (RW,I,LCL,REL,CON)
  $DATAP   037150 000142    (RW,D,LCL,REL,CON)
  OTS$D    037312 000000    (RW,D,LCL,REL,CON)
  OTS$S    037312 000000    (RW,D,LCL,REL,CON)
  SYS$S    037312 000000    (RW,D,LCL,REL,CON)
  $DATA    037312 000150    (RW,D,LCL,REL,CON)
  USER$D   037462 000000    (RW,D,LCL,REL,CON)
  Segment size = 001254 = 342.   words
 Overlay region  000001 Segment 000030
  OTS$I    036206 000236    (RW,I,LCL,REL,CON)
                            DATE    036206
  SYS$I    036444 000142    (RW,I,LCL,REL,CON)
                            TIMASC  036444  TIME    036544
  USER$I   036606 000000    (RW,I,LCL,REL,CON)
  $CODE    036606 002256    (RW,I,LCL,REL,CON)
                            BOUSTA@ 036606
  OTS$O    041064 000000    (RW,I,LCL,REL,CON)
  SYS$O    041064 000000    (RW,I,LCL,REL,CON)
  $DATAP   041064 000406    (RW,D,LCL,REL,CON)
  OTS$D    041472 000000    (RW,D,LCL,REL,CON)
  OTS$S    041472 000000    (RW,D,LCL,REL,CON)
  SYS$S    041472 000000    (RW,D,LCL,REL,CON)
  $DATA    041472 000324    (RW,D,LCL,REL,CON)
  USER$D   042016 000000    (RW,D,LCL,REL,CON)
  Segment size = 003610 = 964.   words
 Overlay region  000001 Segment 000031
  OTS$I    036206 000000    (RW,I,LCL,REL,CON)
  SYS$I    036206 000000    (RW,I,LCL,REL,CON)
  USER$I   036206 000000    (RW,I,LCL,REL,CON)
  $CODE    036206 001272    (RW,I,LCL,REL,CON)
                            BOUTIM@ 036206
  OTS$O    037500 000000    (RW,I,LCL,REL,CON)
  SYS$O    037500 000000    (RW,I,LCL,REL,CON)
  $DATAP   037500 000172    (RW,D,LCL,REL,CON)
  OTS$D    037672 000000    (RW,D,LCL,REL,CON)
  OTS$S    037672 000000    (RW,D,LCL,REL,CON)
  SYS$S    037672 000000    (RW,D,LCL,REL,CON)
  $DATA    037672 000076    (RW,D,LCL,REL,CON)
  USER$D   037770 000000    (RW,D,LCL,REL,CON)
  Segment size = 001562 = 441.   words
 Overlay region  000001 Segment 000032
  OTS$I    036206 000000    (RW,I,LCL,REL,CON)
  SYS$I    036206 000000    (RW,I,LCL,REL,CON)
  USER$I   036206 000000    (RW,I,LCL,REL,CON)
  $CODE    036206 001272    (RW,I,LCL,REL,CON)
                            BOUTMP@ 036206
  OTS$O    037500 000000    (RW,I,LCL,REL,CON)
  SYS$O    037500 000000    (RW,I,LCL,REL,CON)
  $DATAP   037500 000240    (RW,D,LCL,REL,CON)
  OTS$D    037740 000000    (RW,D,LCL,REL,CON)
  OTS$S    037740 000000    (RW,D,LCL,REL,CON)
  SYS$S    037740 000000    (RW,D,LCL,REL,CON)
  $DATA   037740 000022   (RW,D,LCL,REL,CON)
  USER$D  037762 000000   (RW,D,LCL,REL,CON)
  Segment size = 001554 = 438.   words
 Overlay region  000001 Segment 000033
  OTS$I   036206 000000   (RW,I,LCL,REL,CON)
  SYS$I   036206 000000   (RW,I,LCL,REL,CON)
  USER$I  036206 000000   (RW,I,LCL,REL,CON)
  $CODE   036206 000446   (RW,I,LCL,REL,CON)
                           BOUWND@ 036206
  OTS$O   036654 000000   (RW,I,LCL,REL,CON)
  SYS$O   036654 000000   (RW,I,LCL,REL,CON)
  $DATAP  036654 000112   (RW,D,LCL,REL,CON)
  OTS$D   036766 000000   (RW,D,LCL,REL,CON)
  OTS$S   036766 000000   (RW,D,LCL,REL,CON)
  SYS$S   036766 000000   (RW,D,LCL,REL,CON)
  $DATA   036766 000010   (RW,D,LCL,REL,CON)
  USER$D  036776 000000   (RW,D,LCL,REL,CON)
  Segment size = 000570 = 188.   words
 Overlay region  000002 Segment 000034
  OTS$I   042020 000000   (RW,I,LCL,REL,CON)
  SYS$I   042020 000000   (RW,I,LCL,REL,CON)
  USER$I  042020 000000   (RW,I,LCL,REL,CON)
  $CODE   042020 000430   (RW,I,LCL,REL,CON)
                           BOUBIT@ 042020
  OTS$O   042450 000000   (RW,I,LCL,REL,CON)
  SYS$O   042450 000000   (RW,I,LCL,REL,CON)
  $DATAP  042450 000020   (RW,D,LCL,REL,CON)
  OTS$D   042470 000000   (RW,D,LCL,REL,CON)
  OTS$S   042470 000000   (RW,D,LCL,REL,CON)
  SYS$S   042470 000000   (RW,D,LCL,REL,CON)
  $DATA   042470 000060   (RW,D,LCL,REL,CON)
  USER$D  042550 000000   (RW,D,LCL,REL,CON)
  Segment size = 000530 = 172.   words
 Overlay region  000002 Segment 000035
  OTS$I   042020 000000   (RW,I,LCL,REL,CON)
  SYS$I   042020 000000   (RW,I,LCL,REL,CON)
  USER$I  042020 000000   (RW,I,LCL,REL,CON)
  $CODE   042020 000334   (RW,I,LCL,REL,CON)
                           BOUCHN@ 042020
  OTS$O   042354 000000   (RW,I,LCL,REL,CON)
  SYS$O   042354 000000   (RW,I,LCL,REL,CON)
  $DATAP  042354 000106   (RW,D,LCL,REL,CON)
  OTS$D   042462 000000   (RW,D,LCL,REL,CON)
  OTS$S   042462 000000   (RW,D,LCL,REL,CON)
  SYS$S   042462 000000   (RW,D,LCL,REL,CON)
  $DATA   042462 000020   (RW,D,LCL,REL,CON)
  USER$D  042502 000000   (RW,D,LCL,REL,CON)
  Segment size = 000462 = 153.   words
 Overlay region  000002 Segment 000036
  OTS$I   042020 000000   (RW,I,LCL,REL,CON)
  SYS$I   042020 000000   (RW,I,LCL,REL,CON)
  USER$I  042020 000000   (RW,I,LCL,REL,CON)
  $CODE    042020 001414    (RW,I,LCL,REL,CON)
                             TDATE @ 042020
  OTS$O    043434 000000    (RW,I,LCL,REL,CON)
  SYS$O    043434 000000    (RW,I,LCL,REL,CON)
  $DATAP   043434 000014    (RW,D,LCL,REL,CON)
  OTS$D    043450 000000    (RW,D,LCL,REL,CON)
  OTS$S    043450 000000    (RW,D,LCL,REL,CON)
  SYS$S    043450 000000    (RW,D,LCL,REL,CON)
  $DATA    043450 000122    (RW,D,LCL,REL,CON)
  USER$D   043572 000000    (RW,D,LCL,REL,CON)
  Segment size = 001552 = 437.   words
 Overlay region  000002 Segment 000037
  OTS$I    042020 000000    (RW,I,LCL,REL,CON)
  SYS$I    042020 000000    (RW,I,LCL,REL,CON)
  USER$I   042020 000000    (RW,I,LCL,REL,CON)
  $CODE    042020 000352    (RW,I,LCL,REL,CON)
                             DATEI @ 042020
  OTS$O    042372 000000    (RW,I,LCL,REL,CON)
  SYS$O    042372 000000    (RW,I,LCL,REL,CON)
  $DATAP   042372 000014    (RW,D,LCL,REL,CON)
  OTS$D    042406 000000    (RW,D,LCL,REL,CON)
  OTS$S    042406 000000    (RW,D,LCL,REL,CON)
  SYS$S    042406 000000    (RW,D,LCL,REL,CON)
  $DATA    042406 000030    (RW,D,LCL,REL,CON)
  USER$D   042436 000000    (RW,D,LCL,REL,CON)
  Segment size = 000416 = 135.   words
 Overlay region  000002 Segment 000040
  OTS$I    042020 000000    (RW,I,LCL,REL,CON)
  SYS$I    042020 000000    (RW,I,LCL,REL,CON)
  USER$I   042020 000000    (RW,I,LCL,REL,CON)
  $CODE    042020 001136    (RW,I,LCL,REL,CON)
                             BOUDYR@ 042020
  OTS$O    043156 000000    (RW,I,LCL,REL,CON)
  SYS$O    043156 000000    (RW,I,LCL,REL,CON)
  $DATAP   043156 000014    (RW,D,LCL,REL,CON)
  OTS$D    043172 000000    (RW,D,LCL,REL,CON)
  OTS$S    043172 000000    (RW,D,LCL,REL,CON)
  SYS$S    043172 000000    (RW,D,LCL,REL,CON)
  $DATA    043172 000114    (RW,D,LCL,REL,CON)
  USER$D   043306 000000    (RW,D,LCL,REL,CON)
  Segment size = 001266 = 347.   words
 Overlay region  000002 Segment 000041
  OTS$I    042020 000000    (RW,I,LCL,REL,CON)
  SYS$I    042020 000410    (RW,I,LCL,REL,CON)
                             CLOSEC  042020  ICLOSE  042020  IFREEC  042070
                             IGETC   042130  IREAD   042202  IREADW  042210
                             IREADC  042214  $READ   042220  IWRITE  042260
                             IWRITW  042266  IWRITC  042272  $WRITE  042276
                             LOOKUP  042346
  USER$I   042430 000000    (RW,I,LCL,REL,CON)
  $CODE    042430 001000    (RW,I,LCL,REL,CON)
                             BOURED@ 042430
 OTS$O   043430 000000    (RW,I,LCL,REL,CON)
 SYS$O   043430 000000    (RW,I,LCL,REL,CON)
 $DATAP  043430 000126    (RW,D,LCL,REL,CON)
 OTS$D   043556 000000    (RW,D,LCL,REL,CON)
 OTS$S   043556 000000    (RW,D,LCL,REL,CON)
 SYS$S   043556 000000    (RW,D,LCL,REL,CON)
 $DATA   043556 000026    (RW,D,LCL,REL,CON)
 USER$D  043604 000000    (RW,D,LCL,REL,CON)
 Segment size = 001564 = 442.   words
Overlay region  000002 Segment 000042
 OTS$I   042020 000000    (RW,I,LCL,REL,CON)
 SYS$I   042020 000000    (RW,I,LCL,REL,CON)
 USER$I  042020 000000    (RW,I,LCL,REL,CON)
 $CODE   042020 000132    (RW,I,LCL,REL,CON)
                          RET   @ 042020
 OTS$O   042152 000000    (RW,I,LCL,REL,CON)
 SYS$O   042152 000000    (RW,I,LCL,REL,CON)
 $DATAP  042152 000070    (RW,D,LCL,REL,CON)
 OTS$D   042242 000000    (RW,D,LCL,REL,CON)
 OTS$S   042242 000000    (RW,D,LCL,REL,CON)
 SYS$S   042242 000000    (RW,D,LCL,REL,CON)
 $DATA   042242 000004    (RW,D,LCL,REL,CON)
 USER$D  042246 000000    (RW,D,LCL,REL,CON)
 Segment size = 000226 = 75.    words
Overlay region  000002 Segment 000043
 OTS$I   042020 000000    (RW,I,LCL,REL,CON)
 SYS$I   042020 000000    (RW,I,LCL,REL,CON)
 USER$I  042020 000000    (RW,I,LCL,REL,CON)
 $CODE   042020 001044    (RW,I,LCL,REL,CON)
                          RLLTI @ 042020
 OTS$O   043064 000000    (RW,I,LCL,REL,CON)
 SYS$O   043064 000000    (RW,I,LCL,REL,CON)
 $DATAP  043064 000316    (RW,D,LCL,REL,CON)
 OTS$D   043402 000000    (RW,D,LCL,REL,CON)
 OTS$S   043402 000000    (RW,D,LCL,REL,CON)
 SYS$S   043402 000000    (RW,D,LCL,REL,CON)
 $DATA   043402 000024    (RW,D,LCL,REL,CON)
 USER$D  043426 000000    (RW,D,LCL,REL,CON)
 Segment size = 001406 = 387.   words
Overlay region  000002 Segment 000044
 OTS$I   042020 000000    (RW,I,LCL,REL,CON)
 SYS$I   042020 000000    (RW,I,LCL,REL,CON)
 USER$I  042020 000000    (RW,I,LCL,REL,CON)
 $CODE   042020 001366    (RW,I,LCL,REL,CON)
                          RTLLI @ 042020
 OTS$O   043406 000000    (RW,I,LCL,REL,CON)
 SYS$O   043406 000000    (RW,I,LCL,REL,CON)
 $DATAP  043406 000370    (RW,D,LCL,REL,CON)
 OTS$D   043776 000000    (RW,D,LCL,REL,CON)
 OTS$S   043776 000000    (RW,D,LCL,REL,CON)
 SYS$S   043776 000000    (RW,D,LCL,REL,CON)
 $DATA   043776 000030    (RW,D,LCL,REL,CON)
 USER$D  044026 000000    (RW,D,LCL,REL,CON)
 Segment size = 002006 = 515.   words
Overlay region  000002 Segment 000045
 OTS$I   042020 000604    (RW,I,LCL,REL,CON)
                         ALOG10  042020  ALOG    042024  EXP     042206
                         $FCALL  042432  $XFF    042450  XFF$    042462
 SYS$I   042624 000000    (RW,I,LCL,REL,CON)
 USER$I  042624 000000    (RW,I,LCL,REL,CON)
 $CODE   042624 000632    (RW,I,LCL,REL,CON)
                         WRITE @ 042624
 OTS$O   043456 000000    (RW,I,LCL,REL,CON)
 SYS$O   043456 000000    (RW,I,LCL,REL,CON)
 $DATAP  043456 000110    (RW,D,LCL,REL,CON)
 OTS$D   043566 000000    (RW,D,LCL,REL,CON)
 OTS$S   043566 000000    (RW,D,LCL,REL,CON)
 SYS$S   043566 000000    (RW,D,LCL,REL,CON)
 $DATA   043566 000172    (RW,D,LCL,REL,CON)
 USER$D  043760 000000    (RW,D,LCL,REL,CON)
 Segment size = 001740 = 496.   words

Transfer address = 025430, High limit = 044024 = 9226.  words
```

```
RT-11 LINK  V06.01D        Load Map            Mon 12-Sep-83 07:59:14
BOURST.SAV        Title:   BOURST    Ident:    FORV02
Section   Addr    Size     Global    Value     Global    Value     Global    Value

 . ABS.   000000  001000    (RW,I,GBL,ABS,OVR)
                           $USRSW    000000    $RF2A1    000000    .VIR      000002
                           $NLCHN    000006    $HRDWR    000010    $SYSV$    000012
                           $WASIZ    000152    $LRECL    000210
 $OHAND   001000  000106    (RW,I,GBL,REL,CON)
                           $OVRH     001002    O$READ    001024    O$DONE    001036
                           $ODF1     001102    $ODF2     001104
 $OTABL   001106  000266    (RW,D,GBL,REL,OVR)
 OTS$I    001374  020136    (RW,I,LCL,REL,CON)
                           $$OTSI    001374    $CVTDB    001374    $CVTDI    001374
                           $CVTCB    001410    $CVTCI    001410    CIC$      001422
                           CLC$      001422    CID$      001432    CLD$      001432
                           $DI       001432    $CVTFB    001436    $CVTFI    001436
                           CIF$      001450    CLF$      001450    $RI       001450
                           CIL$      001462    CLI$      001466    $CVTIC    001470
                           $CVTID    001470    CCI$      001502    CDI$      001502
                           $IC       001502    $ID       001502    $CVTIF    001506
                           CFI$      001520    $IR       001520    RCI$      001532
                           GCO$      002546    FCO$      002554    ECO$      002560
                           DCO$      002566    ADF$IS    003510    ADF$PS    003524
                           ADF$MS    003536    $ADDF     003546    ADF$SS    003560
                           $ADR      003560    SUF$IS    003570    SUF$PS    003604
                           SUF$MS    003616    $SUBF     003626    SUF$SS    003640
                           $SBR      003640    MUF$MS    003652    MUF$PS    003656
                           MUF$IS    003664    $MULF     003672    MUF$SS    003704
                           $MLR      003704    $OTI      003742    $$OTI     003744
                           $SETOP    004166    $$SET     005774    $INITI    006270
                           $CLOSE    006406    CMF$PS    007164    CMF$MS    007170
                           CMF$IS    007204    $CMPF     007212    CMF$SS    007224
                           $CMR      007224    CMF$PI    007236    CMF$MI    007242
                           CMF$II    007252    CMF$SI    007256    CMF$PP    007270
                           CMF$MP    007274    CMF$IP    007304    CMF$SP    007310
                           CMF$PM    007320    CMF$MM    007324    CMF$IM    007334
                           CMF$SM    007340    OCI$      007372    ICI$      007400
                           $ECI      007414    OCO$      007574    ICO$      007602
                           $DUMPL    010000    $OPNER    010126    $CHKER    010164
                           $IOEXI    010210    $EOL      010256    EOL$      010260
                           $ERRTB    010374    $ERRS     010501    EXIT      014242
                           $FCHNL    014246    $FIO      015110    $$FIO     015114
                           IDATE     016260    IFW$      016372    $IFW      016376
                           $$IFW     016402    IFW$$     016440    ILW$      016510
                           $ILW      016514    TVS$      016634    $TVS      016636
                           $PUTRE    017450    $PUTBL    017756    $GETBL    020166
                           $EOFIL    020352    $EOF2     020366    SAVRG$    020406
                           THRD$     020564    $STPS     020566    STP$      020574
                           $STP      020574    FOO$      020600    $EXIT     020620
                           $OTIS     020744    $$OTIS    020746    TVL$      021066
                           $TVL      021066    TVF$      021074    $TVF      021074
```

```
                          TVD$     021102  $TVD     021102  TVQ$     021110
                          $TVQ     021110  TVP$     021116  $TVP     021116
                          TVI$     021124  $TVI     021124  $WAIT    021260
                          $VRINT   021322
 OTS$P    021532 000054    (RW,D,GBL,REL,OVR)
 SYS$I    021606 000622    (RW,I,LCL,REL,CON)
                          CVTTIM   021606  $DIVTK   021650  $DIV60   021676
                          GTIM     021742  IPEEK    021762  IPOKE    021772
                          IPEEKB   022002  IPOKEB   022016  ISLEEP   022026
                          ISPY     022066  $GVAL    022072  ITTINR   022110
                          ITTOUR   022120  SCCA     022134  ISCOMP   022172
                          SCOMP    022172  SETCMD   022262  $TIME$   022314
 USER$I   022430 000000    (RW,I,LCL,REL,CON)
 $CODE    022430 003604    (RW,I,LCL,REL,CON)
                          $$OTSC   022430  SYSTIM   026034
 OTS$O    026234 001066    (RW,I,LCL,REL,CON)
                          $$OTSO   026234  $OPEN    026234
 SYS$O    027322 000000    (RW,I,LCL,REL,CON)
 $DATAP   027322 001262    (RW,D,LCL,REL,CON)
 OTS$D    030604 000006    (RW,D,LCL,REL,CON)
                          NHCLN$   030610
 OTS$S    030612 000002    (RW,D,LCL,REL,CON)
                          $AOTS    030612
 SYS$S    030614 000004    (RW,D,LCL,REL,CON)
                          $SYSLB   030614  $LOCK    030616  $CRASH   030617
 $DATA    030620 001014    (RW,D,LCL,REL,CON)
 USER$D   031634 000000    (RW,D,LCL,REL,CON)
 .$$$$.   031634 000000    (RW,D,GBL,REL,OVR)
 SPACE    031634 000050    (RW,D,GBL,REL,OVR)
 Segment size = 031704 = 6626.   words
Overlay region  000001 Segment 000001
 OTS$I    031706 000000    (RW,I,LCL,REL,CON)
 SYS$I    031706 000000    (RW,I,LCL,REL,CON)
 USER$I   031706 000000    (RW,I,LCL,REL,CON)
 $CODE    031706 000706    (RW,I,LCL,REL,CON)
                          BOUHDR@ 031706
 OTS$O    032614 000000    (RW,I,LCL,REL,CON)
 SYS$O    032614 000000    (RW,I,LCL,REL,CON)
 $DATAP   032614 000444    (RW,D,LCL,REL,CON)
 OTS$D    033260 000000    (RW,D,LCL,REL,CON)
 OTS$S    033260 000000    (RW,D,LCL,REL,CON)
 SYS$S    033260 000000    (RW,D,LCL,REL,CON)
 $DATA    033260 000000    (RW,D,LCL,REL,CON)
 USER$D   033260 000000    (RW,D,LCL,REL,CON)
 Segment size = 001352 = 373.    words
Overlay region  000001 Segment 000002
 OTS$I    031706 000000    (RW,I,LCL,REL,CON)
 SYS$I    031706 000000    (RW,I,LCL,REL,CON)
 USER$I   031706 000000    (RW,I,LCL,REL,CON)
 $CODE    031706 000440    (RW,I,LCL,REL,CON)
                          BOUFLL@ 031706
 OTS$O    032346 000000    (RW,I,LCL,REL,CON)
 SYS$O   032346 000000   (RW,I,LCL,REL,CON)
 $DATAP  032346 000232   (RW,D,LCL,REL,CON)
 OTS$D   032600 000000   (RW,D,LCL,REL,CON)
 OTS$S   032600 000000   (RW,D,LCL,REL,CON)
 SYS$S   032600 000000   (RW,D,LCL,REL,CON)
 $DATA   032600 000010   (RW,D,LCL,REL,CON)
 USER$D  032610 000000   (RW,D,LCL,REL,CON)
 Segment size = 000702 = 225.   words
Overlay region  000001 Segment 000003
 OTS$I   031706 000000   (RW,I,LCL,REL,CON)
 SYS$I   031706 000410   (RW,I,LCL,REL,CON)
                       CLOSEC  031706  ICLOSE  031706  IFREEC  031756
                       IGETC   032016  IREAD   032070  IREADW  032076
                       IREADC  032102  $READ   032106  IWRITE  032146
                       IWRITW  032154  IWRITC  032160  $WRITE  032164
                       LOOKUP  032234
 USER$I  032316 000000   (RW,I,LCL,REL,CON)
 $CODE   032316 001000   (RW,I,LCL,REL,CON)
                       BOURED@ 032316
 OTS$O   033316 000000   (RW,I,LCL,REL,CON)
 SYS$O   033316 000000   (RW,I,LCL,REL,CON)
 $DATAP  033316 000126   (RW,D,LCL,REL,CON)
 OTS$D   033444 000000   (RW,D,LCL,REL,CON)
 OTS$S   033444 000000   (RW,D,LCL,REL,CON)
 SYS$S   033444 000000   (RW,D,LCL,REL,CON)
 $DATA   033444 000026   (RW,D,LCL,REL,CON)
 USER$D  033472 000000   (RW,D,LCL,REL,CON)
 DIAG    033472 000300   (RW,D,GBL,REL,OVR)
 Segment size = 002064 = 538.   words
Overlay region  000001 Segment 000004
 OTS$I   031706 000274   (RW,I,LCL,REL,CON)
                       $GETRE  031706  $TTYIN  031762  IFR$    032116
                       $IFR    032122  $$IFR   032126  IFR$$   032160
 SYS$I   032202 000000   (RW,I,LCL,REL,CON)
 USER$I  032202 000000   (RW,I,LCL,REL,CON)
 $CODE   032202 002652   (RW,I,LCL,REL,CON)
                       BOUHLP@ 032202
 OTS$O   035054 000000   (RW,I,LCL,REL,CON)
 SYS$O   035054 000000   (RW,I,LCL,REL,CON)
 $DATAP  035054 002762   (RW,D,LCL,REL,CON)
 OTS$D   040036 000000   (RW,D,LCL,REL,CON)
 OTS$S   040036 000000   (RW,D,LCL,REL,CON)
 SYS$S   040036 000000   (RW,D,LCL,REL,CON)
 $DATA   040036 000004   (RW,D,LCL,REL,CON)
 USER$D  040042 000000   (RW,D,LCL,REL,CON)
 Segment size = 006134 = 1582.  words
Overlay region  000002 Segment 000005
 OTS$I   040044 000000   (RW,I,LCL,REL,CON)
 SYS$I   040044 000000   (RW,I,LCL,REL,CON)
 USER$I  040044 000000   (RW,I,LCL,REL,CON)
 $CODE   040044 000040   (RW,I,LCL,REL,CON)
                       VTDHLB@ 040044
 OTS$O   040104 000000   (RW,I,LCL,REL,CON)
 SYS$O   040104 000000   (RW,I,LCL,REL,CON)
 $DATAP  040104 000026   (RW,D,LCL,REL,CON)
 OTS$D   040132 000000   (RW,D,LCL,REL,CON)
 OTS$S   040132 000000   (RW,D,LCL,REL,CON)
 SYS$S   040132 000000   (RW,D,LCL,REL,CON)
 $DATA   040132 000000   (RW,D,LCL,REL,CON)
 USER$D  040132 000000   (RW,D,LCL,REL,CON)
 Segment size = 000066 = 27.    words
Overlay region  000002 Segment 000006
 OTS$I   040044 000000   (RW,I,LCL,REL,CON)
 SYS$I   040044 000000   (RW,I,LCL,REL,CON)
 USER$I  040044 000000   (RW,I,LCL,REL,CON)
 $CODE   040044 000040   (RW,I,LCL,REL,CON)
                          VTDHLT@ 040044
 OTS$O   040104 000000   (RW,I,LCL,REL,CON)
 SYS$O   040104 000000   (RW,I,LCL,REL,CON)
 $DATAP  040104 000026   (RW,D,LCL,REL,CON)
 OTS$D   040132 000000   (RW,D,LCL,REL,CON)
 OTS$S   040132 000000   (RW,D,LCL,REL,CON)
 SYS$S   040132 000000   (RW,D,LCL,REL,CON)
 $DATA   040132 000000   (RW,D,LCL,REL,CON)
 USER$D  040132 000000   (RW,D,LCL,REL,CON)
 Segment size = 000066 = 27.    words
Overlay region  000002 Segment 000007
 OTS$I   040044 000000   (RW,I,LCL,REL,CON)
 SYS$I   040044 000000   (RW,I,LCL,REL,CON)
 USER$I  040044 000000   (RW,I,LCL,REL,CON)
 $CODE   040044 000200   (RW,I,LCL,REL,CON)
                          VTED  @ 040044
 OTS$O   040244 000000   (RW,I,LCL,REL,CON)
 SYS$O   040244 000000   (RW,I,LCL,REL,CON)
 $DATAP  040244 000036   (RW,D,LCL,REL,CON)
 OTS$D   040302 000000   (RW,D,LCL,REL,CON)
 OTS$S   040302 000000   (RW,D,LCL,REL,CON)
 SYS$S   040302 000000   (RW,D,LCL,REL,CON)
 $DATA   040302 000002   (RW,D,LCL,REL,CON)
 USER$D  040304 000000   (RW,D,LCL,REL,CON)
 Segment size = 000240 = 80.    words
Overlay region  000002 Segment 000010
 OTS$I   040044 000000   (RW,I,LCL,REL,CON)
 SYS$I   040044 000000   (RW,I,LCL,REL,CON)
 USER$I  040044 000000   (RW,I,LCL,REL,CON)
 $CODE   040044 000130   (RW,I,LCL,REL,CON)
                          VTERSE@ 040044
 OTS$O   040174 000000   (RW,I,LCL,REL,CON)
 SYS$O   040174 000000   (RW,I,LCL,REL,CON)
 $DATAP  040174 000034   (RW,D,LCL,REL,CON)
 OTS$D   040230 000000   (RW,D,LCL,REL,CON)
 OTS$S   040230 000000   (RW,D,LCL,REL,CON)
 SYS$S   040230 000000   (RW,D,LCL,REL,CON)
 $DATA   040230 000000   (RW,D,LCL,REL,CON)
```

```
  USER$D  040230 000000   (RW,D,LCL,REL,CON)
  Segment size = 000164 = 58.    words
 Overlay region  000002 Segment 000011
  OTS$I   040044 000000   (RW,I,LCL,REL,CON)
  SYS$I   040044 000000   (RW,I,LCL,REL,CON)
  USER$I  040044 000000   (RW,I,LCL,REL,CON)
  $CODE   040044 000770   (RW,I,LCL,REL,CON)
                          VTHVP @ 040044
  OTS$O   041034 000000   (RW,I,LCL,REL,CON)
  SYS$O   041034 000000   (RW,I,LCL,REL,CON)
  $DATAP  041034 000146   (RW,D,LCL,REL,CON)
  OTS$D   041202 000000   (RW,D,LCL,REL,CON)
  OTS$S   041202 000000   (RW,D,LCL,REL,CON)
  SYS$S   041202 000000   (RW,D,LCL,REL,CON)
  $DATA   041202 000004   (RW,D,LCL,REL,CON)
  USER$D  041206 000000   (RW,D,LCL,REL,CON)
  Segment size = 001142 = 305.   words
 Overlay region  000002 Segment 000012
  OTS$I   040044 000000   (RW,I,LCL,REL,CON)
  SYS$I   040044 000000   (RW,I,LCL,REL,CON)
  USER$I  040044 000000   (RW,I,LCL,REL,CON)
  $CODE   040044 001334   (RW,I,LCL,REL,CON)
                          VTSGR @ 040044
  OTS$O   041400 000000   (RW,I,LCL,REL,CON)
  SYS$O   041400 000000   (RW,I,LCL,REL,CON)
  $DATAP  041400 000040   (RW,D,LCL,REL,CON)
  OTS$D   041440 000000   (RW,D,LCL,REL,CON)
  OTS$S   041440 000000   (RW,D,LCL,REL,CON)
  SYS$S   041440 000000   (RW,D,LCL,REL,CON)
  $DATA   041440 000010   (RW,D,LCL,REL,CON)
  USER$D  041450 000000   (RW,D,LCL,REL,CON)
  Segment size = 001404 = 386.   words
 Overlay region  000002 Segment 000013
  OTS$I   040044 000000   (RW,I,LCL,REL,CON)
  SYS$I   040044 000000   (RW,I,LCL,REL,CON)
  USER$I  040044 000000   (RW,I,LCL,REL,CON)
  $CODE   040044 000434   (RW,I,LCL,REL,CON)
                          VTTBM @ 040044
  OTS$O   040500 000000   (RW,I,LCL,REL,CON)
  SYS$O   040500 000000   (RW,I,LCL,REL,CON)
  $DATAP  040500 000072   (RW,D,LCL,REL,CON)
  OTS$D   040572 000000   (RW,D,LCL,REL,CON)
  OTS$S   040572 000000   (RW,D,LCL,REL,CON)
  SYS$S   040572 000000   (RW,D,LCL,REL,CON)
  $DATA   040572 000004   (RW,D,LCL,REL,CON)
  USER$D  040576 000000   (RW,D,LCL,REL,CON)
  Segment size = 000532 = 173.   words
 Overlay region  000002 Segment 000014
  OTS$I   040044 000000   (RW,I,LCL,REL,CON)
  SYS$I   040044 000000   (RW,I,LCL,REL,CON)
  USER$I  040044 000000   (RW,I,LCL,REL,CON)
  $CODE   040044 000114   (RW,I,LCL,REL,CON)
```

```
                          VTWID @ 040044
 OTS$O    040160 000000   (RW,I,LCL,REL,CON)
 SYS$O    040160 000000   (RW,I,LCL,REL,CON)
 $DATAP   040160 000042   (RW,D,LCL,REL,CON)
 OTS$D    040222 000000   (RW,D,LCL,REL,CON)
 OTS$S    040222 000000   (RW,D,LCL,REL,CON)
 SYS$S    040222 000000   (RW,D,LCL,REL,CON)
 $DATA    040222 000002   (RW,D,LCL,REL,CON)
 USER$D   040224 000000   (RW,D,LCL,REL,CON)
 Segment size = 000160 = 56.    words
Overlay region  000002 Segment 000015
 OTS$I    040044 000000   (RW,I,LCL,REL,CON)
 SYS$I    040044 000000   (RW,I,LCL,REL,CON)
 USER$I   040044 000000   (RW,I,LCL,REL,CON)
 $CODE    040044 000040   (RW,I,LCL,REL,CON)
                          VTDWL @ 040044
 OTS$O    040104 000000   (RW,I,LCL,REL,CON)
 SYS$O    040104 000000   (RW,I,LCL,REL,CON)
 $DATAP   040104 000026   (RW,D,LCL,REL,CON)
 OTS$D    040132 000000   (RW,D,LCL,REL,CON)
 OTS$S    040132 000000   (RW,D,LCL,REL,CON)
 SYS$S    040132 000000   (RW,D,LCL,REL,CON)
 $DATA    040132 000000   (RW,D,LCL,REL,CON)
 USER$D   040132 000000   (RW,D,LCL,REL,CON)
 Segment size = 000066 = 27.    words

Transfer address = 022430, High limit = 041446 = 8595.  words
```

```
RT-11 LINK  V06.01D     Load Map        Wed 28-Sep-83 08:06:00
BOUFIL.SAV      Title:  BOUFIM  Ident:  FORV02
Section  Addr   Size    Global  Value   Global  Value   Global  Value

 . ABS.  000000 001000  (RW,I,GBL,ABS,OVR)
                        $USRSW  000000  $RF2A1  000000  .VIR    000002
                        $NLCHN  000006  $HRDWR  000010  $SYSV$  000012
                        $WASIZ  000152  $LRECL  000210
 $OHAND  001000 000106  (RW,I,GBL,REL,CON)
                        $OVRH   001002  O$READ  001024  O$DONE  001036
                        $ODF1   001102  $ODF2   001104
 $OTABL  001106 000336  (RW,D,GBL,REL,OVR)
 OTS$I   001444 022722  (RW,I,LCL,REL,CON)
                        $$OTSI  001444  $CVTDB  001444  $CVTDI  001444
                        $CVTCB  001460  $CVTCI  001460  CIC$    001472
                        CLC$    001472  CID$    001502  CLD$    001502
                        $DI     001502  $CVTFB  001506  $CVTFI  001506
                        CIF$    001520  CLF$    001520  $RI     001520
                        CIL$    001532  CLI$    001536  $CVTIC  001540
                        $CVTID  001540  CCI$    001552  CDI$    001552
                        $IC     001552  $ID     001552  $CVTIF  001556
                        CFI$    001570  $IR     001570  RCI$    001602
                        GCO$    002616  FCO$    002624  ECO$    002630
                        DCO$    002636  ADF$IS  003560  ADF$PS  003574
                        ADF$MS  003606  $ADDF   003616  ADF$SS  003630
                        $ADR    003630  SUF$IS  003640  SUF$PS  003654
                        SUF$MS  003666  $SUBF   003676  SUF$SS  003710
                        $SBR    003710  MUF$MS  003722  MUF$PS  003726
                        MUF$IS  003734  $MULF   003742  MUF$SS  003754
                        $MLR    003754  $OTI    004012  $$OTI   004014
                        $SETOP  004236  $$SET   006044  $INITI  006340
                        ABS     006456  ASSIGN  006474  NMI$1M  007120
                        NMI$1I  007132  BLE$    007142  BEQ$    007144
                        BGT$    007152  BGE$    007154  BRA$    007156
                        BNE$    007162  BLT$    007164  CAI$    007174
                        CAL$    007202  $CLOSE  007232  CLOSE   010010
                        CMF$PS  010034  CMF$MS  010040  CMF$IS  010054
                        $CMPF   010062  CMF$SS  010074  $CMR    010074
                        CMF$PI  010106  CMF$MI  010112  CMF$II  010122
                        CMF$SI  010126  CMF$PP  010140  CMF$MP  010144
                        CMF$IP  010154  CMF$SP  010160  CMF$PM  010170
                        CMF$MM  010174  CMF$IM  010204  CMF$SM  010210
                        OCI$    010242  ICI$    010250  $ECI    010264
                        OCO$    010444  ICO$    010452  DATE    010650
                        $DUMPL  011106  END$    011234  ERR$    011246
                        $END    011260  $ERR    011276  $OPNER  011320
                        $CHKER  011356  $IOEXI  011402  $EOL    011450
                        EOL$    011452  $ERRTB  011566  $ERRS   011673
                        EXIT    015434  $FCHNL  015440  $FIO    016302
                        $$FIO   016306  $GETFI  017452  $GETRE  017510
                        $TTYIN  017564  ADI$SS  017720  ADI$SA  017724
                        ADI$SM  017730  ADI$IS  017734  ADI$IA  017740
```

```
                         ADI$IM   017744   ADI$MS   017750   ADI$MA   017754
                         ADI$MM   017760   IDATE    017764   IFR$     020076
                         $IFR     020102   $$IFR    020106   IFR$$    020140
                         IFW$     020162   $IFW     020166   $$IFW    020172
                         IFW$$    020230   ILW$     020300   $ILW     020304
                         TVS$     020424   $TVS     020426   MOI$SS   021240
                         MOL$SS   021240   MOI$SM   021244   MOI$SA   021250
                         MOI$IS   021254   MOL$IS   021254   REL$     021254
                         MOI$IM   021260   MOI$IA   021264   MOI$MS   021270
                         MOI$MM   021274   MOI$MA   021300   MOI$0S   021304
                         MOI$0M   021310   MOI$0A   021314   MOI$1S   021320
                         MOI$1M   021326   MOI$1A   021334   ICI$S    021342
                         ICI$M    021346   ICI$P    021352   ICI$A    021354
                         DCI$S    021360   DCI$M    021364   DCI$P    021370
                         DCI$A    021372   CMI$IP   021376   CMI$SP   021400
                         CMI$MP   021406   CMI$PP   021416   CMI$PS   021422
                         CMI$PI   021430   CMI$PM   021436   MOI$IP   021444
                         MOI$SP   021446   MOI$PP   021454   MOI$MP   021460
                         MOI$PS   021470   MOI$PM   021476   MOI$PA   021504
                         MOI$0P   021512   MOI$1P   021520   MOL$SM   021530
                         MOL$SA   021534   MOL$MS   021540   MOL$MM   021550
                         MOL$MA   021554   MOL$SP   021560   MOL$PP   021566
                         MOL$MP   021572   MOL$PM   021602   MOL$PS   021610
                         MOL$PA   021614   MOL$IM   021622   MOL$IA   021630
                         MOL$IP   021636   NMI$1P   021646   NPI$1P   021664
                         NPI$1I   021670   NPI$1M   021704   $PUTRE   021726
                         RET$L    022234   RET$F    022240   RET$I    022246
                         RET$     022250   $PUTBL   022304   $GETBL   022514
                         $EOFIL   022700   $EOF2    022714   SAVRG$   022734
                         THRD$    023112   $STPS    023114   STP$     023122
                         $STP     023122   FOO$     023126   $EXIT    023146
                         $OTIS    023272   $$OTIS   023274   TAI$     023414
                         $TAI     023414   TAL$     023422   $TAL     023422
                         TAQ$     023430   $TAQ     023430   TAP$     023436
                         $TAP     023436   TAD$     023444   $TAD     023444
                         TAF$     023452   $TAF     023452   TVL$     023674
                         $TVL     023674   TVF$     023702   $TVF     023702
                         TVD$     023710   $TVD     023710   TVQ$     023716
                         $TVQ     023716   TVP$     023724   $TVP     023724
                         TVI$     023732   $TVI     023732   $WAIT    024066
                         $VRINT   024130   SAL$IP   024340   SAL$SP   024342
                         SVL$IP   024346   SVL$SP   024350   SAL$MP   024356
                         SVL$MP   024362
OTS$P     024366 000054   (RW,D,GBL,REL,OVR)
SYS$I     024442 001240   (RW,I,LCL,REL,CON)
                         CLOSEC   024442   ICLOSE   024442   $DIVTK   024512
                         $DIV60   024540   IFREEC   024604   IGETC    024644
                         IPEEK    024716   IPOKE    024726   IPEEKB   024736
                         IPOKEB   024752   ISLEEP   024762   ISPY     025022
                         $GVAL    025026   IWRITE   025044   IWRITW   025052
                         IWRITC   025056   $WRITE   025062   LOOKUP   025132
                         R50ASC   025214   SETCMD   025334   TIMASC   025366
```

```
                        TIME    025466  $TIME$  025530  TRIM    025644
 USER$I  025702 000000   (RW,I,LCL,REL,CON)
 $CODE   025702 002476   (RW,I,LCL,REL,CON)
                        $$OTSC  025702  CLRLIN  030102  CLRBOT  030142
                        GETSTR  030170
 OTS$O   030400 001066   (RW,I,LCL,REL,CON)
                        $$OTSO  030400  $OPEN   030400
 SYS$O   031466 000000   (RW,I,LCL,REL,CON)
 $DATAP  031466 000762   (RW,D,LCL,REL,CON)
 OTS$D   032450 000006   (RW,D,LCL,REL,CON)
                        NHCLN$  032454
 OTS$S   032456 000002   (RW,D,LCL,REL,CON)
                        $AOTS   032456
 SYS$S   032460 000004   (RW,D,LCL,REL,CON)
                        $SYSLB  032460  $LOCK   032462  $CRASH  032463
 $DATA   032464 000160   (RW,D,LCL,REL,CON)
 USER$D  032644 000000   (RW,D,LCL,REL,CON)
 .$$$$.  032644 000000   (RW,D,GBL,REL,OVR)
 FILE    032644 000024   (RW,D,GBL,REL,OVR)
 LOCAT   032670 000012   (RW,D,GBL,REL,OVR)
 DATA    032702 000730   (RW,D,GBL,REL,OVR)
 Segment size = 033632 = 7117.   words
Overlay region  000001 Segment 000001
 OTS$I   033634 000000   (RW,I,LCL,REL,CON)
 SYS$I   033634 000000   (RW,I,LCL,REL,CON)
 USER$I  033634 000000   (RW,I,LCL,REL,CON)
 $CODE   033634 003456   (RW,I,LCL,REL,CON)
                        SCRNRD@ 033634
 OTS$O   037312 000000   (RW,I,LCL,REL,CON)
 SYS$O   037312 000000   (RW,I,LCL,REL,CON)
 $DATAP  037312 001140   (RW,D,LCL,REL,CON)
 OTS$D   040452 000000   (RW,D,LCL,REL,CON)
 OTS$S   040452 000000   (RW,D,LCL,REL,CON)
 SYS$S   040452 000000   (RW,D,LCL,REL,CON)
 $DATA   040452 000110   (RW,D,LCL,REL,CON)
 USER$D  040562 000000   (RW,D,LCL,REL,CON)
 Segment size = 004726 = 1259.   words
Overlay region  000001 Segment 000002
 OTS$I   033634 000000   (RW,I,LCL,REL,CON)
 SYS$I   033634 000000   (RW,I,LCL,REL,CON)
 USER$I  033634 000000   (RW,I,LCL,REL,CON)
 $CODE   033634 001516   (RW,I,LCL,REL,CON)
                        READF @ 033634
 OTS$O   035352 000000   (RW,I,LCL,REL,CON)
 SYS$O   035352 000000   (RW,I,LCL,REL,CON)
 $DATAP  035352 000622   (RW,D,LCL,REL,CON)
 OTS$D   036174 000000   (RW,D,LCL,REL,CON)
 OTS$S   036174 000000   (RW,D,LCL,REL,CON)
 SYS$S   036174 000000   (RW,D,LCL,REL,CON)
 $DATA   036174 000072   (RW,D,LCL,REL,CON)
 USER$D  036266 000000   (RW,D,LCL,REL,CON)
 Segment size = 002432 = 653.   words
```

```
Overlay region  000001 Segment 000003
 OTS$I    033634 000000   (RW,I,LCL,REL,CON)
 SYS$I    033634 000000   (RW,I,LCL,REL,CON)
 USER$I   033634 000000   (RW,I,LCL,REL,CON)
 $CODE    033634 001020   (RW,I,LCL,REL,CON)
                        BOUFID@ 033634
 OTS$O    034654 000000   (RW,I,LCL,REL,CON)
 SYS$O    034654 000000   (RW,I,LCL,REL,CON)
 $DATAP   034654 000436   (RW,D,LCL,REL,CON)
 OTS$D    035312 000000   (RW,D,LCL,REL,CON)
 OTS$S    035312 000000   (RW,D,LCL,REL,CON)
 SYS$S    035312 000000   (RW,D,LCL,REL,CON)
 $DATA    035312 000054   (RW,D,LCL,REL,CON)
 USER$D   035366 000000   (RW,D,LCL,REL,CON)
 Segment size = 001532 = 429.   words
Overlay region  000001 Segment 000004
 OTS$I    033634 000000   (RW,I,LCL,REL,CON)
 SYS$I    033634 000000   (RW,I,LCL,REL,CON)
 USER$I   033634 000000   (RW,I,LCL,REL,CON)
 $CODE    033634 000542   (RW,I,LCL,REL,CON)
                        BHELP @ 033634
 OTS$O    034376 000000   (RW,I,LCL,REL,CON)
 SYS$O    034376 000000   (RW,I,LCL,REL,CON)
 $DATAP   034376 001320   (RW,D,LCL,REL,CON)
 OTS$D    035716 000000   (RW,D,LCL,REL,CON)
 OTS$S    035716 000000   (RW,D,LCL,REL,CON)
 SYS$S    035716 000000   (RW,D,LCL,REL,CON)
 $DATA    035716 000006   (RW,D,LCL,REL,CON)
 USER$D   035724 000000   (RW,D,LCL,REL,CON)
 Segment size = 002070 = 540.   words
Overlay region  000001 Segment 000005
 OTS$I    033634 000000   (RW,I,LCL,REL,CON)
 SYS$I    033634 000000   (RW,I,LCL,REL,CON)
 USER$I   033634 000000   (RW,I,LCL,REL,CON)
 $CODE    033634 000742   (RW,I,LCL,REL,CON)
                        WRITE @ 033634
 OTS$O    034576 000000   (RW,I,LCL,REL,CON)
 SYS$O    034576 000000   (RW,I,LCL,REL,CON)
 $DATAP   034576 000472   (RW,D,LCL,REL,CON)
 OTS$D    035270 000000   (RW,D,LCL,REL,CON)
 OTS$S    035270 000000   (RW,D,LCL,REL,CON)
 SYS$S    035270 000000   (RW,D,LCL,REL,CON)
 $DATA    035270 000042   (RW,D,LCL,REL,CON)
 USER$D   035332 000000   (RW,D,LCL,REL,CON)
 Segment size = 001476 = 415.   words
Overlay region  000001 Segment 000006
 OTS$I    033634 000000   (RW,I,LCL,REL,CON)
 SYS$I    033634 000000   (RW,I,LCL,REL,CON)
 USER$I   033634 000000   (RW,I,LCL,REL,CON)
 $CODE    033634 001360   (RW,I,LCL,REL,CON)
                        PARTL @ 033634
 OTS$O    035214 000000   (RW,I,LCL,REL,CON)
 SYS$O    035214 000000    (RW,I,LCL,REL,CON)
 $DATAP   035214 000516    (RW,D,LCL,REL,CON)
 OTS$D    035732 000000    (RW,D,LCL,REL,CON)
 OTS$S    035732 000000    (RW,D,LCL,REL,CON)
 SYS$S    035732 000000    (RW,D,LCL,REL,CON)
 $DATA    035732 000060    (RW,D,LCL,REL,CON)
 USER$D   036012 000000    (RW,D,LCL,REL,CON)
 Segment size = 002156 = 567.   words
Overlay region  000001 Segment 000007
 OTS$I    033634 000000    (RW,I,LCL,REL,CON)
 SYS$I    033634 000610    (RW,I,LCL,REL,CON)
                          IDSTAT   033634  IENTER   033652  IFETCH   033740
                          IQSET    034042  IRAD50   034140  RAD50    034176
                          ISCOMP   034354  SCOMP    034354
 USER$I   034444 000000    (RW,I,LCL,REL,CON)
 $CODE    034444 001544    (RW,I,LCL,REL,CON)
                          NEWDSK@ 034444
 OTS$O    036210 000000    (RW,I,LCL,REL,CON)
 SYS$O    036210 000000    (RW,I,LCL,REL,CON)
 $DATAP   036210 001754    (RW,D,LCL,REL,CON)
 OTS$D    040164 000000    (RW,D,LCL,REL,CON)
 OTS$S    040164 000000    (RW,D,LCL,REL,CON)
 SYS$S    040164 000000    (RW,D,LCL,REL,CON)
 $DATA    040164 001104    (RW,D,LCL,REL,CON)
 USER$D   041270 000000    (RW,D,LCL,REL,CON)
 Segment size = 005434 = 1422.  words
Overlay region  000001 Segment 000010
 OTS$I    033634 000000    (RW,I,LCL,REL,CON)
 SYS$I    033634 000000    (RW,I,LCL,REL,CON)
 USER$I   033634 000000    (RW,I,LCL,REL,CON)
 $CODE    033634 000466    (RW,I,LCL,REL,CON)
                          FILELB@ 033634
 OTS$O    034322 000000    (RW,I,LCL,REL,CON)
 SYS$O    034322 000000    (RW,I,LCL,REL,CON)
 $DATAP   034322 001102    (RW,D,LCL,REL,CON)
 OTS$D    035424 000000    (RW,D,LCL,REL,CON)
 OTS$S    035424 000000    (RW,D,LCL,REL,CON)
 SYS$S    035424 000000    (RW,D,LCL,REL,CON)
 $DATA    035424 000000    (RW,D,LCL,REL,CON)
 USER$D   035424 000000    (RW,D,LCL,REL,CON)
 Segment size = 001570 = 444.   words
Overlay region  000002 Segment 000011
 OTS$I    041272 001444    (RW,I,LCL,REL,CON)
                          $CVTFC   041272  $CVTFD   041272  CCF$     041304
                          CDF$     041304  $RC      041304  $RD      041304
                          $CVTDF   041316  CFD$     041330  $DR      041330
                          SUD$PS   041346  SUD$MS   041354  SUD$IS   041366
                          $SUBD    041400  SUD$SS   041412  $SBD     041412
                          ADD$PS   041430  ADD$MS   041436  ADD$IS   041450
                          $ADDD    041462  ADD$SS   041474  $ADD     041474
                          DEXP     041510  DLOG10   042054  DLOG     042060
                          MUD$MS   042344  MUD$PS   042352  MUD$IS   042362
```

```
                              $MULD    042374  MUD$SS   042406  $MLD     042406
                              $FCALL   042422  $XFD     042440  XFD$     042452
                              $XDF     042504  XDF$     042516  $XDD     042530
                              XDD$     042542
  SYS$I     042736 000000      (RW, I,LCL, REL,CON)
  USER$I    042736 000000      (RW, I,LCL, REL,CON)
  $CODE     042736 001166      (RW, I,LCL, REL,CON)
                              CONVRT@ 042736
  OTS$O     044124 000000      (RW, I,LCL, REL,CON)
  SYS$O     044124 000000      (RW, I,LCL, REL,CON)
  $DATAP    044124 000020      (RW, D,LCL, REL,CON)
  OTS$D     044144 000000      (RW, D,LCL, REL,CON)
  OTS$S     044144 000000      (RW, D,LCL, REL,CON)
  SYS$S     044144 000000      (RW, D,LCL, REL,CON)
  $DATA     044144 000456      (RW, D,LCL, REL,CON)
  USER$D    044622 000000      (RW, D,LCL, REL,CON)
  Segment size = 003330 = 876.    words
 Overlay region  000002 Segment 000012
  OTS$I     041272 000000      (RW, I,LCL, REL,CON)
  SYS$I     041272 000000      (RW, I,LCL, REL,CON)
  USER$I    041272 000000      (RW, I,LCL, REL,CON)
  $CODE     041272 000040      (RW, I,LCL, REL,CON)
                              VTDHLB  041272
  OTS$O     041332 000000      (RW, I,LCL, REL,CON)
  SYS$O     041332 000000      (RW, I,LCL, REL,CON)
  $DATAP    041332 000026      (RW, D,LCL, REL,CON)
  OTS$D     041360 000000      (RW, D,LCL, REL,CON)
  OTS$S     041360 000000      (RW, D,LCL, REL,CON)
  SYS$S     041360 000000      (RW, D,LCL, REL,CON)
  $DATA     041360 000000      (RW, D,LCL, REL,CON)
  USER$D    041360 000000      (RW, D,LCL, REL,CON)
  Segment size = 000066 = 27.     words
 Overlay region  000002 Segment 000013
  OTS$I     041272 000000      (RW, I,LCL, REL,CON)
  SYS$I     041272 000000      (RW, I,LCL, REL,CON)
  USER$I    041272 000000      (RW, I,LCL, REL,CON)
  $CODE     041272 000040      (RW, I,LCL, REL,CON)
                              VTDHLT  041272
  OTS$O     041332 000000      (RW, I,LCL, REL,CON)
  SYS$O     041332 000000      (RW, I,LCL, REL,CON)
  $DATAP    041332 000026      (RW, D,LCL, REL,CON)
  OTS$D     041360 000000      (RW, D,LCL, REL,CON)
  OTS$S     041360 000000      (RW, D,LCL, REL,CON)
  SYS$S     041360 000000      (RW, D,LCL, REL,CON)
  $DATA     041360 000000      (RW, D,LCL, REL,CON)
  USER$D    041360 000000      (RW, D,LCL, REL,CON)
  Segment size = 000066 = 27.     words
 Overlay region  000002 Segment 000014
  OTS$I     041272 000000      (RW, I,LCL, REL,CON)
  SYS$I     041272 000000      (RW, I,LCL, REL,CON)
  USER$I    041272 000000      (RW, I,LCL, REL,CON)
  $CODE     041272 000200      (RW, I,LCL, REL,CON)
```

```
                             VTED  @ 041272
  OTS$O    041472 000000   (RW,I,LCL,REL,CON)
  SYS$O    041472 000000   (RW,I,LCL,REL,CON)
  $DATAP   041472 000036   (RW,D,LCL,REL,CON)
  OTS$D    041530 000000   (RW,D,LCL,REL,CON)
  OTS$S    041530 000000   (RW,D,LCL,REL,CON)
  SYS$S    041530 000000   (RW,D,LCL,REL,CON)
  $DATA    041530 000002   (RW,D,LCL,REL,CON)
  USER$D   041532 000000   (RW,D,LCL,REL,CON)
  Segment size = 000240 = 80.    words
 Overlay region  000002 Segment 000015
  OTS$I    041272 000000   (RW,I,LCL,REL,CON)
  SYS$I    041272 000000   (RW,I,LCL,REL,CON)
  USER$I   041272 000000   (RW,I,LCL,REL,CON)
  $CODE    041272 000200   (RW,I,LCL,REL,CON)
                             VTEL  @ 041272
  OTS$O    041472 000000   (RW,I,LCL,REL,CON)
  SYS$O    041472 000000   (RW,I,LCL,REL,CON)
  $DATAP   041472 000036   (RW,D,LCL,REL,CON)
  OTS$D    041530 000000   (RW,D,LCL,REL,CON)
  OTS$S    041530 000000   (RW,D,LCL,REL,CON)
  SYS$S    041530 000000   (RW,D,LCL,REL,CON)
  $DATA    041530 000002   (RW,D,LCL,REL,CON)
  USER$D   041532 000000   (RW,D,LCL,REL,CON)
  Segment size = 000240 = 80.    words
 Overlay region  000002 Segment 000016
  OTS$I    041272 000000   (RW,I,LCL,REL,CON)
  SYS$I    041272 000000   (RW,I,LCL,REL,CON)
  USER$I   041272 000000   (RW,I,LCL,REL,CON)
  $CODE    041272 000130   (RW,I,LCL,REL,CON)
                             VTERSE@ 041272
  OTS$O    041422 000000   (RW,I,LCL,REL,CON)
  SYS$O    041422 000000   (RW,I,LCL,REL,CON)
  $DATAP   041422 000034   (RW,D,LCL,REL,CON)
  OTS$D    041456 000000   (RW,D,LCL,REL,CON)
  OTS$S    041456 000000   (RW,D,LCL,REL,CON)
  SYS$S    041456 000000   (RW,D,LCL,REL,CON)
  $DATA    041456 000000   (RW,D,LCL,REL,CON)
  USER$D   041456 000000   (RW,D,LCL,REL,CON)
  Segment size = 000164 = 58.    words
 Overlay region  000002 Segment 000017
  OTS$I    041272 000000   (RW,I,LCL,REL,CON)
  SYS$I    041272 000000   (RW,I,LCL,REL,CON)
  USER$I   041272 000000   (RW,I,LCL,REL,CON)
  $CODE    041272 000770   (RW,I,LCL,REL,CON)
                             VTHVP @ 041272
  OTS$O    042262 000000   (RW,I,LCL,REL,CON)
  SYS$O    042262 000000   (RW,I,LCL,REL,CON)
  $DATAP   042262 000146   (RW,D,LCL,REL,CON)
  OTS$D    042430 000000   (RW,D,LCL,REL,CON)
  OTS$S    042430 000000   (RW,D,LCL,REL,CON)
  SYS$S    042430 000000   (RW,D,LCL,REL,CON)
 $DATA   042430 000004    (RW,D,LCL,REL,CON)
 USER$D  042434 000000    (RW,D,LCL,REL,CON)
 Segment size = 001142 = 305.   words
Overlay region  000002 Segment 000020
 OTS$I   041272 000000    (RW,I,LCL,REL,CON)
 SYS$I   041272 000000    (RW,I,LCL,REL,CON)
 USER$I  041272 000000    (RW,I,LCL,REL,CON)
 $CODE   041272 001334    (RW,I,LCL,REL,CON)
                         VTSGR @ 041272
 OTS$O   042626 000000    (RW,I,LCL,REL,CON)
 SYS$O   042626 000000    (RW,I,LCL,REL,CON)
 $DATAP  042626 000040    (RW,D,LCL,REL,CON)
 OTS$D   042666 000000    (RW,D,LCL,REL,CON)
 OTS$S   042666 000000    (RW,D,LCL,REL,CON)
 SYS$S   042666 000000    (RW,D,LCL,REL,CON)
 $DATA   042666 000010    (RW,D,LCL,REL,CON)
 USER$D  042676 000000    (RW,D,LCL,REL,CON)
 Segment size = 001404 = 386.   words
Overlay region  000002 Segment 000021
 OTS$I   041272 000000    (RW,I,LCL,REL,CON)
 SYS$I   041272 000056    (RW,I,LCL,REL,CON)
                         IREAD   041272  IREADW  041300  IREADC  041304
                         $READ   041310
 USER$I  041350 000000    (RW,I,LCL,REL,CON)
 $CODE   041350 001000    (RW,I,LCL,REL,CON)
                         BOURED@ 041350
 OTS$O   042350 000000    (RW,I,LCL,REL,CON)
 SYS$O   042350 000000    (RW,I,LCL,REL,CON)
 $DATAP  042350 000126    (RW,D,LCL,REL,CON)
 OTS$D   042476 000000    (RW,D,LCL,REL,CON)
 OTS$S   042476 000000    (RW,D,LCL,REL,CON)
 SYS$S   042476 000000    (RW,D,LCL,REL,CON)
 $DATA   042476 000026    (RW,D,LCL,REL,CON)
 USER$D  042524 000000    (RW,D,LCL,REL,CON)
 DIAG    042524 000300    (RW,D,GBL,REL,OVR)
 Segment size = 001532 = 429.   words

Transfer address = 025702, High limit = 044620 = 9416.  words
```

APPENDIX VI

COMMAND FILES TO BUILD SYSTEM

```
! +------------------------------------------------------------------+
! |     File:     TIMLIB.COM                                         |
! |     Function: Command file to build timlib.obj from libtim.for   !
! |                  library resident on SY:                         |
! |                                                                  |
! +------------------------------------------------------------------+
!
FORT/NOLINE/OBJ:OUP: INP:LIBTIM
LIB/CREATE SY:TIMLIB OUP:LIBTIM
```

```
! +--------------------------------------------------------------------+
! |     File:     BOULDA.COM                                           |
! |     Function: Command file build BOULD.REL on device BOU:          |
! |               BOULDF.COM  compiles fortran programs                |
! |               BOULDL.COM  links compiled file into a task file     |
! |                                                                    |
! +--------------------------------------------------------------------+
@BOULDF
@BOULDL
```

```
! +-----------------------------------------------------------------------+
! |     File:     BOULDF.COM                                              |
! |     Function: Command file to fortran source codes for BOULD.REL      |
! |                                                                       |
! +-----------------------------------------------------------------------+
!
!       set terminal echo off
!
SET TT QUIET
!
!       main subroutines
!
FORT/NOLINE/OBJ:OUP: INP:BOULDM.FOR
FORT/NOLINE/OBJ:OUP: INP:BOUCHK.FOR
FORT/NOLINE/OBJ:OUP: INP:BOUCHN.FOR
FORT/NOLINE/OBJ:OUP: INP:BOUCON.FOR
FORT/NOLINE/OBJ:OUP: INP:BOUDAC.FOR
FORT/NOLINE/OBJ:OUP: INP:BOUDAL.FOR
FORT/NOLINE/OBJ:OUP: INP:BOUDAT.FOR
FORT/NOLINE/OBJ:OUP: INP:BOUDK1.FOR
FORT/NOLINE/OBJ:OUP: INP:BOUDK2.FOR
FORT/NOLINE/OBJ:OUP: INP:BOUER1.FOR
FORT/NOLINE/OBJ:OUP: INP:BOUER2.FOR
FORT/NOLINE/OBJ:OUP: INP:BOUFAL.FOR
FORT/NOLINE/OBJ:OUP: INP:BOUFFT.FOR
FORT/NOLINE/OBJ:OUP: INP:BOUIN1.FOR
FORT/NOLINE/OBJ:OUP: INP:BOUIN2.FOR
FORT/NOLINE/OBJ:OUP: INP:BOUIN3.FOR
FORT/NOLINE/OBJ:OUP: INP:BOUMHT.FOR
FORT/NOLINE/OBJ:OUP: INP:BOUMH1.FOR
FORT/NOLINE/OBJ:OUP: INP:BOUMH2.FOR
FORT/NOLINE/OBJ:OUP: INP:BOUOSC.FOR
FORT/NOLINE/OBJ:OUP: INP:BOUPDC.FOR
FORT/NOLINE/OBJ:OUP: INP:BOUPFT.FOR
FORT/NOLINE/OBJ:OUP: INP:BOUPKC.FOR
FORT/NOLINE/OBJ:OUP: INP:BOUPMP.FOR
FORT/NOLINE/OBJ:OUP: INP:BOUPRS.FOR
FORT/NOLINE/OBJ:OUP: INP:BOUSLP.FOR
FORT/NOLINE/OBJ:OUP: INP:BOUSTA.FOR
FORT/NOLINE/OBJ:OUP: INP:BOUTIM.FOR
FORT/NOLINE/OBJ:OUP: INP:BOUTMP.FOR
FORT/NOLINE/OBJ:OUP: INP:BOUWND.FOR
!
!       secondary subroutines
!
FORT/NOLINE/OBJ:OUP: INP:BOUBIT.FOR
FORT/NOLINE/OBJ:OUP: INP:BOUDTC.FOR
FORT/NOLINE/OBJ:OUP: INP:BOUDTI.FOR
FORT/NOLINE/OBJ:OUP: INP:BOUDYR.FOR
FORT/NOLINE/OBJ:OUP: INP:BOURED.FOR
FORT/NOLINE/OBJ:OUP: INP:BOURET.FOR
FORT/NOLINE/OBJ:OUP: INP:BOURLL.FOR
```

```
FORT/NOLINE/OBJ:OUP: INP:BOURTL.FOR
FORT/NOLINE/OBJ:OUP: INP:BOUWRT.FOR
!
!       turn echo back on
!
SET TT NOQUIET
```

```
! +-----------------------------------------------------------------------+
! |      File:     BOULDL.COM                                              |
! |      Function: Command file to link object files for BOULD.REL         |
! |                 Produces foreground program                            |
! |                                                                        |
! +-----------------------------------------------------------------------+
!
!       turn off echo on terminal
!
SET TT QUIET
!
!       create .REL file and .MAP file
!          TIMLIB is a library of time calculations
!
R LINK
BOU:BOULD,OUP:BOULD=OUP:BOULDM,OUP:BOUDAT,SY:TIMLIB/R:400/C
!
!       main subroutines
!
OUP:BOUCHK/O:1/C
OUP:BOUCON/O:1/C
OUP:BOUDAC/O:1/C
OUP:BOUDAL/O:1/C
OUP:BOUDK1/O:1/C
OUP:BOUDK2/O:1/C
OUP:BOUER1/O:1/C
OUP:BOUER2/O:1/C
OUP:BOUFAL/O:1/C
OUP:BOUFFT/O:1/C
OUP:BOUIN1/O:1/C
OUP:BOUIN2/O:1/C
OUP:BOUIN3/O:1/C
OUP:BOUMH1/O:1/C
OUP:BOUMH2/O:1/C
OUP:BOUOSC/O:1/C
OUP:BOUMHT/O:1/C
OUP:BOUPDC/O:1/C
OUP:BOUPFT/O:1/C
OUP:BOUPKC/O:1/C
OUP:BOUPMP/O:1/C
OUP:BOUPRS/O:1/C
OUP:BOUSLP/O:1/C
OUP:BOUSTA/O:1/C
OUP:BOUTIM/O:1/C
OUP:BOUTMP/O:1/C
OUP:BOUWND/O:1/C
!
!       secondary or control subroutines
!
OUP:BOUBIT/O:2/C
OUP:BOUCHN/O:2/C
OUP:BOUDTC/O:2/C
```

```
OUP:BOUDTI/O:2/C
OUP:BOUDYR/O:2/C
OUP:BOURED/O:2/C
OUP:BOURET/O:2/C
OUP:BOURLL/O:2/C
OUP:BOURTL/O:2/C
OUP:BOUWRT/O:2
C
```

```
! +--------------------------------------------------------------------+
! |    File:       BOURSL.COM                                          |
! |    Function:   Command file to link BOURST routines                |
! |                program is linked into an overlay sections due to size |
! |                                                                    |
! +--------------------------------------------------------------------+
!
R LINK
!
!       link driver and timlib
!
BOU:BOURST,OUP:BOURST=OUP:BOURST,SY:TIMLIB/C
!
!       main subroutines
!
OUP:BOUHDR/O:1/C
OUP:BOUFLL/O:1/C
OUP:BOURED/O:1/C
OUP:BOUHLP/O:1/C
!
!       video handlers
!
OUP:VTDHLB/O:2/C
OUP:VTDHLT/O:2/C
OUP:VTED/O:2/C
OUP:VTERSE/O:2/C
OUP:VTHVP/O:2/C
OUP:VTSGR/O:2/C
OUP:VTTBM/O:2/C
OUP:VTWID/O:2/C
OUP:VTDWL/O:2
C
```

```
! +======================================================================+
! |     File:     BOURSA.COM                                             |
! |     Function: Command file build BOURST.SAV                          |
! |               BOURSF.COM  compiles the fortran files                 |
! |               BOURSL.COM  links the compiled fortran programs in to  |
! |                a task file                                           |
! |                                                                      |
! +----------------------------------------------------------------------+
@BOURSF
@BOURSL
```

```
! +---------------------------------------------------------------------+
! |     File:     BOUFIA.COM                                            |
! |     Function: Command file build BOUFIL.SAV                         |
! |                  File management program                            |
! |                                                                     |
! |                BOUFIF.COM  compiles source files                    |
! |                BOUFIL.COM  links compiled files into task file      |
! |                                                                     |
! +---------------------------------------------------------------------+
!
!
@BOUFIF
@BOUFIL
```

```
! +----------------------------------------------------------------------+
! |     File:     BOURSF.COM                                             !
! |     Function: Command file to fortran files for BOURST.SAV           !
! |                                                                      !
! +----------------------------------------------------------------------+
!
!       main driver
!
FORT/NOLINE/OBJ:OUP: INP:BOURST
!
!       paints screen with labels
!
FORT/NOLINE/OBJ:OUP: INP:BOUHDR
!
!       program failure routine
!
FORT/NOLINE/OBJ:OUP: INP:BOUFLL
!
!       disk reading section (common with bould.rel)
!
FORT/NOLINE/OBJ:OUP: INP:BOURED
!
!       help file
!
FORT/NOLINE/OBJ:OUP: INP:BOUHLP
!
!       double heigth line bottom
!
FORT/NOLINE/OBJ:OUP: INP:VTDHLB
!
!       double heigth line top
!
FORT/NOLINE/OBJ:OUP: INP:VTDHLT
!
!       erase sections of display
!
FORT/NOLINE/OBJ:OUP: INP:VTED
!
!       erase all display and return to home
!
FORT/NOLINE/OBJ:OUP: INP:VTERSE
!
!       horizontal and vertical positioning
!
FORT/NOLINE/OBJ:OUP: INP:VTHVP
!
!       selecte grapih attributes (reverse image etc.)
!
FORT/NOLINE/OBJ:OUP: INP:VTSGR
!
!       set top and bottom margins
!
```

```
FORT/NOLINE/OBJ:OUP: INP:VTTBM
!
!       double width line
!
FORT/NOLINE/OBJ:OUP: INP:VTDWL
!
!       screen width (80-132)
!
FORT/NOLINE/OBJ:OUP: INP:VTWID
```

```
! +------------------------------------------------------------------------+
! |     File:     BOURSL.COM                                               |
! |     Function: Command file link object modules for BOURST.SAV          |
! |                                                                        |
! +------------------------------------------------------------------------+
!
R LINK
!
!       link driver and timlib
!
BOU:BOURST,OUP:BOURST=OUP:BOURST,SY:TIMLIB/C
!
!       main subroutines
!
OUP:BOUHDR/O:1/C
OUP:BOUFLL/O:1/C
OUP:BOURED/O:1/C
OUP:BOUHLP/O:1/C
!
!       video handlers
!
OUP:VTDHLB/O:2/C
OUP:VTDHLT/O:2/C
OUP:VTED/O:2/C
OUP:VTERSE/O:2/C
OUP:VTHVP/O:2/C
OUP:VTSGR/O:2/C
OUP:VTTBM/O:2/C
OUP:VTWID/O:2/C
OUP:VTDWL/O:2
C
```

```
! +-------------------------------------------------------------------------+
! |     File:     BOUFIF.COM                                                |
! |     Function: Command file to fortran files for BOUFIL.SAV              |
! +-------------------------------------------------------------------------+
!
!
!main driver for file maintenence
!
FORT/NOLINE/OBJ:OUP: INP:BOUFIM
!
!reinitialization file
!
FORT/NOLINE/OBJ:OUP: INP:BOUFIW
!
!partial output file
!
FORT/NOLINE/OBJ:OUP: INP:BOUFIP
!
!new files created
!
FORT/NOLINE/OBJ:OUP: INP:BOUFIN
!
!file to read data
!
FORT/NOLINE/OBJ:OUP: INP:BOUFIR
!
!file header information
!
FORT/NOLINE/OBJ:OUP: INP:BOUFIF
!
!read seleced records
!
FORT/NOLINE/OBJ:OUP: INP:BOUFIS
!
!dump data for inform
!
FORT/NOLINE/OBJ:OUP: INP:BOUFID
!
!help file
!
FORT/NOLINE/OBJ:OUP: INP:BOUFIH
!
!ascii conversion
!
FORT/NOLINE/OBJ:OUP: INP:CONVRT
!
!part of vtlib
!
FORT/NOLINE/OBJ:OUP: INP:VTEL
!
```

```
! +-----------------------------------------------------------------+
! |     File:     BOUFIL.COM                                        |
! |     Function: Command file TO link object modules for BOUFIL.SAV |
! |                                                                 |
! +-----------------------------------------------------------------+
!
!
!       link maintence file
!
R LINK
BOU:BOUFIL,OUP:BOUFIL=OUP:BOUFIM/C
OUP:BOUFIS/O:1/C
OUP:BOUFIR/O:1/C
OUP:BOUFID/O:1/C
OUP:BOUFIH/O:1/C
OUP:BOUFIW/O:1/C
OUP:BOUFIP/O:1/C
OUP:BOUFIN/O:1/C
OUP:BOUFIF/O:1/C
!
!       ASCII conversion to real*4
!
OUP:CONVRT/O:2/C
!
!       screen handlers
!
OUP:VTDHLB/O:2/C
OUP:VTDHLT/O:2/C
OUP:VTED/O:2/C
OUP:VTEL/O:2/C
OUP:VTERSE/O:2/C
OUP:VTHVP/O:2/C
OUP:VTSGR/O:2/C
!
!       file read/write file
!
OUP:BOURED/O:2
C
```

```
! +-----------------------------------------------------------------------+
! |     File:     BOULD.COM                                               |
! |     Function: Command file start boulder program                      |
! |                                                                       |
! +-----------------------------------------------------------------------+
!
!       start boulder program
!
FRUN BOU:BOULD.REL/BUFFER:2500/TERM:1
```

```
! +-------------------------------------------------------------------------+
! |     File:     RESTRT.COM                                                |
! |     Function: Command file to restart boulder program                  |
! |                                                                         |
! +-------------------------------------------------------------------------+
!
RESET
!
!       close all output files
!
CLOSE
!
!       start boulder
!
@BOULD
!
!       start status program
!
RU BOU:BOURST
```

```
! +------------------------------------------------------------------------+
! |     File:     BFILE.COM                                                |
! |     Function: Command file to start BOUFIL management program         |
! |               from BOURST                                              |
! |                                                                        |
! +------------------------------------------------------------------------+
!
!
RU BOU:BOUFIL
```

```
! +----------------------------------------------------------------+
! |     File:     BSTRT.COM                                         |
! |     Function: Command file to start BOURST management program   |
! |                   from BOUFIL                                   |
! |                                                                 |
! +----------------------------------------------------------------+
!
RU BOU:BOURST
```

APPENDIX VII

FUNCTIONS FOR TIME AND DATE CALCULATIONS

```
c+------------------------------------------------------------------------+
c|  Program : BOULD.REL,BOURST,BOUFIL                                      |
c|  Module  : LIBTIM                                                       |
c|  Function: SOURCE FILES FOR TIME FUNCTIONS                              |
c+------------------------------------------------------------------------+
c|              System time calculation library                           |
c|                                                                        |
c|        This library converts time value to real*4 values which can    |
c|                   be stored in the disk files                          |
c|                                                                        |
c|         all values must be stored as real*4 because of file structure|
c|                                                                        |
c|       function call:                                                   |
c|                                                                        |
c|           TIMDIF - calcultes time differences                          |
c|           SYSTIM - returns system time and date as real*4 values      |
c|           SYSTMC - returns decimal equivalent of the time              |
c|           TIMADD - adds two times together                             |
c|                                                                        |
c+------------------------------------------------------------------------+
      FUNCTION TIMDIF(TIME1,TIME2)
c--
c--   calculates the differences in two times
c--   put in order new time , old time
c--
      IF (TIME1.LT.TIME2) TIME1=TIME1+2400.
c--
c--   determine difference
c--
      DIFF=TIME1-TIME2
      IDIFF=DIFF
c--
c--   calculate numer of hours
c--
      IDIFF=(IDIFF/100)*100
c--
c--   calculate number of minutes
c--
      DIFFM=DIFF-IDIFF
c--
c--   correct minutes for minutes > 60
c--
      IF (DIFFM.GE.40.) TIMDIF=IDIFF+(DIFFM-40.)
c--
      IF (DIFFM.LT.40.) TIMDIF=IDIFF+DIFFM
      RETURN
      END
c------------------------------------------------------------------------
c--
c--
      FUNCTION SYSTIM(SYSDAT)
c--
```

```
c--     returns date and time as real*4 values
c--
        INTEGER*4 ITIME
c--
c--     get system date in integer format
c--       ID1=month  ID2=day   ID3=year
c--
        CALL IDATE(IMN,IDY,IYR)
c--
c--     system date converted to real*4
c--
        SYSDAT=(FLOAT(IYR)*10000.)+(FLOAT(IMN)*100.)+(FLOAT(IDY))
c--
c--     get time i*4 format
c--
        CALL GTIM(ITIME)
c--
c--     convert to integer format
c--
        CALL CVTTIM(ITIME,IHRS,IMIN,ISEC,ITCK)
c--
c--     system time in r*4 format
c--
        SYSTIM=(IHRS*100)+IMIN
        RETURN
        END
c-----------------------------------------------------------------------
c--
c--
        FUNCTION SYSTMC(ZTIME)
c--
c--     returns time as real*4 values which is decimal equivalent
c--     of minutes  ex  9:30 =  9.50  ztime is the current hour
c--
c--
        INTEGER*4 ITIME
c--
c--     get system time  i*4 format
c--
        CALL GTIM(ITIME)
c--
c--     convert to i*2 format
c--
        CALL CVTTIM(ITIME,IHRS,IMIN,ISEC,ITCK)
c--
c--     convert i*2 to r*4 for decimal math
c--
        HRS=IHRS
        XMIN=IMIN
        XMIN=XMIN/60.
        ZTIME=HRS
c--
```

```
c--     add hours and decimal minutes
c--
        SYSTMC=HRS+XMIN
        RETURN
        END
c-----------------------------------------------------------------------
c--
c--
        FUNCTION TIMADD(T1,T2,IF)
c--
c--     add two times together
c--      remember ther are 60 minutes to an hour and can not be added diry
c--     IF flag that a new day has occurred
c--
c--     set flag to 0  no new day
c--
        IF=0
c--
c--     determine hours
c--
        IH1=T1/100
        IH2=T2/100
c--
c--     total hours
c--
        IHT=IH1+IH2
c--
c--     convert minutes
c--
        IM1=T1-(IH1*100)
        IM2=T2-(IH2*100)
c--
c--     total minutes
c--
        IMT=IM1+IM2
c--
c--     is this greater than minutes in hour?
c--             yes - inc hour and correct minutes
c--             no  - branch
c--
        IF (IMT.LT.60) GOTO 10
        IHT=IHT+1
        IMT=IMT-60
c--
c--     is this a new day?
c--             yes - correct and set flag
c--             no  - branch
c--
10      IF (IHT.LT.24) GOTO 20
        IHT=IHT-24
        IF=1
c--
```

```
c--     correct time
c--
20      TIMADD=(IHT*100)+IMT
c--
        RETURN
        END
```